UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

E.A.P. DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

Algoritmo del banquero : aplicado al sistema visado de poderes caso BBVA Banco Continental

TESINA

Para optar el Título de Ingeniero de Sistemas

AUTOR

Marlon Brañez Reyes

LIMA – PERÚ
2011

FICHA CATALOGRÁFICA

BRAÑEZ REYES, Marlon

ALGORITMO DEL BANQUERO: APLICADO AL SISTEMA VISADO DE PODERES CASO BBVA BANCO CONTINENTAL

Ingeniería de software
(Lima, Perú 2011)

Tesina, Facultad de Ingeniería de Sistemas, Pregrado, Universidad Nacional Mayor De San Marcos

Formato 28 x 20 cm Paginas #

DEDICATORIA:

Este trabajo está dedicado a toda mi familia en especial a mi madre.

AGRADECIMIENTOS

Al Magister Percy De la Cruz Velez de Villa, por su orientación y dedicación para que este trabajo cumpla con los objetivos trazados.

A los profesores de la UNMSM, por sus observaciones teóricas que me sirvieron de mucho.

A todas aquellas personas que indirectamente me ayudaron para culminar este trabajo y que muchas veces constituyen un invalorable apoyo.

Y por encima de todo doy gracias a Dios.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERIA DE SISTEMAS E INFORMÁTICA

ESCUELA ACADEMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ALGORITMO DEL BANQUERO: APLICADO AL SISTEMA VISADO DE PODERES CASO BBVA BANCO CONTINENTAL

Autor: BRAÑEZ REYES, Marlon

Asesor: DE LA CRUZ VELEZ DE VILLA, Percy

Titulo: Tesina, para optar el Título Profesional de Ingeniero de Sistemas

Fecha: Mayo de 2011

RESUMEN

En la actualidad el mantener una ventaja competitiva, especialmente orientada al cliente, sobre los competidores es algo imprescindible; esto se puede lograr optimizando procesos y siendo eficientes en la asignación de recursos. Optimizando procesos iniciados por clientes así mismo reduciendo los tiempos de respuesta, del cual dependen o se ven involucrados los clientes; permitirá mantener la cartera de clientes o captar potenciales clientes.

El proceso en estudio es el Visado de Poderes de la entidad financiera BBVA Continental, para lo cual optimizaremos la gestión del trámite documentario mediante el Sistema de Visado de Poderes. Mejorando su gestión en las asignaciones de los recursos disponibles del departamento de

servicios jurídicos de la entidad financiera, en la cual se va implantar el Sistema de Visado de Poderes.

El proceso de visado de poderes consiste en validar ciertos documentos dependiendo del caso, las personas encargadas de validar estos documentos son abogados de distintos estudios de Jurídicos, para ello se tiene que hacer llegar estos documentos para su validación lo cual es realizado por mensajería interna, una vez que estos documentos son revisados se procede a dar un veredicto, aprobado o rechazado, después del veredicto estos documentos retornan a la oficina de origen, donde fueron inicialmente recepcionados para ser visados; todo el ir y venir de los documentos así mismo la distribución de la carga laboral en el departamento legal, hace que el tiempo de la repuesta del la cual está a la espera el cliente no sea el optimo.

El tiempo de respuesta del proceso de visado de poderes se verá reducido debido a la automatización de proceso así como a la implementación del Algoritmo del Banquero en la asignación de recursos o distribución de carga la laboral.

Palabras claves: Visado de Poderes, Asignación de Recursos, Algoritmo del Banquero.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERIA DE SISTEMAS E INFORMÁTICA

ESCUELA ACADEMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ALGORITMO DEL BANQUERO: APLICADO AL SISTEMA VISADO DE PODERES CASO BBVA BANCO CONTINENTAL

Autor: BRAÑEZ REYES, Marlon

Asesor: DE LA CRUZ VELEZ DE VILLA, Percy

Titulo: Tesina, para optar el Título Profesional de Ingeniero de Sistemas

Fecha: Mayo de 2011

ABSTRACT

Currently maintaining a competitive edge, customer-focused particularly on competitors is a must; this can be achieved by optimizing processes and being efficient in allocating resources. Optimizing processes initiated by customers while simultaneously reducing response times, which depend upon or clients are involved, will maintain the customer base or attract potential customers.

The process under discussion is the endorsement of the bank branches BBVA Continental, for which optimize the management of the documentary process through the Visa System Powers. Improving their management in the allocation of available resources of the legal services department of the bank, which will implement the Visa System Powers.

The visa process is to validate certain powers depending on the case documents, the persons authorized to validate these documents are different attorneys Legal studies, this will have to get these documents for validation which is done by internal mail, once these documents are reviewed proceeds to give a verdict, approved or rejected after the verdict, these documents are returned to the office of origin, where they were initially Front Desk for visas, all the comings and goings of the papers the same distribution of the workload in the legal department makes the response time of which awaits the customer is not the optimum.

The response time of the visa process of power will be reduced due to automation of process and the implementation of Banker's Algorithm in the allocation of resources or the work load distribution.

Keywords: Visa Credentials, Resource Allocation, Banker's Algorithm.

ÍNDICE DE CONTENIDOS

Lista de figuras

Lista de Tablas

Introducción

La decisión de elaborar una **“Tesis sobre Algoritmo del Banquero: aplicado al Sistema visado de Poderes”** surgió con el objetivo de dar respuesta a las expectativas despertadas en la agilización de los distintos tramites documentarios así mismo la optimización de la distribución de carga de trabajo entre los recursos competentes.

TODOS TENEMOS NUESTRO ARCHIVO PARTICULAR y estamos acostumbrados a conservar –con más o menos orden– los documentos esenciales que hacen valer nuestros derechos, y otros que testimonian nuestra actividad y trayectoria personal.

Un fondo documental que constituye nuestra memoria, fundamentalmente en soporte papel, y que está en pleno período de transformación a consecuencia de todo un conjunto de acontecimientos que englobamos bajo la denominación de la nueva era digital. [5]

La asignación de recursos es la distribución de activos productivos en sus diferentes usos.

El asunto de la asignación de recursos, se origina de como las sociedades buscan balancear los recursos limitados como el capital, el trabajo y la tierra, frente a las diversas e ilimitadas necesidades de sus integrantes. Los mecanismos de asignación de recursos abarcan el sistema de precios en las economías de libre mercado y la planeación gubernamental, ya sea en las economías operadas por el estado o en el sector público de economías mixtas.

La finalidad de distribuir los recursos es siempre la de obtener la máxima productividad posible a partir de una combinación dada de activos. [7]

Una asignación óptima de los Factores Productivos es aquella que consigue producir la mayor cantidad de Bienes con el mínimo de Recursos, lo que implica canalizar los Recursos productivos hacia aquellas actividades con las más altas rentabilidades. [8]

CAPÍTULO I. PLANTEAMIENTO METODOLÓGICO

1.1 Antecedentes del problema

Nos encontramos en los tiempos de la digitalización de documentos, comercio electrónico, gobierno electrónico el cual nos lleva rescatar la forma de cómo gestionar los procesos de tramites documentarios así mismo la importancia de la digitalización de documentos y sus ventajas sobre los documentos físicos o reales .

La distribución de cargas de trabajo tiene antecedentes en la asignación de recursos en la rama de las Matemáticas, Investigación Operativa, mediante la programación lineal y sus aplicaciones; existen otras técnicas y estudios para la distribución de cargas de trabajo que también tienen antecedentes en la asignación de recursos como en el método Húngaro, algoritmo genético, etc. Algunos de ellos se irán describiendo posteriormente en el Estado del Arte.

1.2 Definición o formulación del problema

EL tiempo de respuesta de la evaluación de solicitudes a visar por el departamento legal depende mucho del tiempo que demore mensajería interna en hacer llegar los documentos a visar al departamento legal, sumado el tiempo de distribución de las solicitudes así mismo el tiempo utilizado por los recursos del departamento en dar un dictamen sobre las solicitudes.

La falta de una forma adecuada de distribución de trabajo; la distribución de carga laboral en el departamento de servicios jurídicos se maneja de una manera poco adecuada pues carece de criterio alguno más que el personal.

Una correcta distribución de la carga de trabajo evitara la sobrecarga laboral de los recursos del departamento de servicios jurídicos, teniendo en cuenta parámetros para distribuir esta carga de trabajo como el tipo de solicitud a evaluar, la experiencia del recurso a asignar la solicitud, etc.

El proceso actual de visado de poderes consiste generalmente en hacer llegar los documentos correspondientes, de acuerdo al trámite a realizar, al área legal para la verificación de los documentos adjuntos al servicio brindado o al tramite a realizar, una

vez validados los documentos retornan a la oficina de origen, con las observaciones correspondientes dependiendo si ha sido rechazado o aprobado. Este es el procedimiento regular en el cual utilizan como medio de distribución de los documentos a mensajería interna, el cual se encarga de hacer llegar los documentos de una oficina a otra. Existen excepciones en cual por agilizar el trámite se envía dichos documentos escaneados por correo electrónico para su verificación.

Los recursos empleados para la verificación de los documentos adjuntos y validación del trámite es un abogado de los distintos estudios con el que cuenta el área legal.

A continuación se muestra una tabla estadística con algunos de los servicios brindados en cual nos permite tener una media del tiempo que lleva realizar estos trámites.

Tramites	Documentos necesarios por tramite	Tiempo promedio en hacer llegar los documentos al área legal	Tiempo promedio en asignar un trámite a un recurso (abogado)	Tiempo promedio en retornar los documentos a la oficina de origen	Tiempo promedio empleado por recurso para validar el tramite	Tiempo total promedio empleado para dar una respuesta sobre el tramite realizado en días
REGISTRO EN EL SISTEMA DE APODERADO	✓ COPIA DNI ✓ ORIGINAL PARTIDA	1 - 3 días	5 - 10 minutos	1 - 3 días	5 - 10 minutos	2 - 6 días
CLIENTES FALLECIDOS CON TESTAMENTO	✓ ORIGINAL DE LA PARTIDA ✓ CARTA DE SOLICITUD ✓ COPIA DEL TESTAMENTO ✓ COPIA DNI	1 - 3 días	5 - 10 minutos	1 - 3 días	20 - 30 minutos	2 - 6 días
CLIENTES FALLECIDOS SIN TESTAMENTO	✓ COPIA DNI	1 - 3 días	5 - 10 minutos	1 - 3 días	5 minutos	2 - 6 días

Tabla 1 Cuadro estadístico anterior a la implementación

El en cuadro anterior nos permite identificar que el cuello de botella es en la distribución de los documentos, realizado por mensajería interna, otro factor a tomar en cuenta es el tiempo empleado en asignar los tramites a los recursos (abogados).

1.3 Objetivos

1.3.1 Objetivo general

La automatización del proceso de visado de poderes mediante el desarrollo del **Sistema Visado de Poderes aplicando el Algoritmo del Banquero** el cual permitirá la agilización del tramite documentario del visado de poderes así como optimizar el desempeño del departamento legal.

1.3.2 Objetivos específicos

- ✓ Analizar del proceso de visado de poderes
- ✓ Implementar algoritmo del banquero
- ✓ Crear componentes que permitan la carga de archivos al sistema
- ✓ Desarrollar e implementar el Sistema Visado de Poderes
- ✓ Reducir el tiempo de respuesta del proceso de Visado de Poderes
- ✓ Establecer un criterio de asignación de recursos

1.4 Justificación

La globalización y el entorno altamente competitivo exigen a las empresas estar orientadas al mercado y gestionar su cartera de clientes como un activo estratégico clave para aumentar la rentabilidad del negocio y lograr una ventaja competitiva sostenible en el tiempo. [11]

En la actualidad mantener un ventaja competitiva sobre los competidores orientada al cliente es primordial, ya sea agilizando procesos, reduciendo tiempo de respuesta de procesos, especialmente procesos que son iniciados por los potenciales clientes.

No solamente con la automatización del proceso se espera agilizar y reducir el tiempo de respuesta del proceso de visado de poderes sino también mediante una mejora en la asignación de recursos del área que se encarga de resolver, visar los documentos respectivos.

El Sistema Visado de Poderes nos servirá:

- ✓ Para tener una Gestión eficiente de los Expedientes
- ✓ Para hacer seguimiento a los Expedientes
- ✓ Para saber cuáles son los documentos que te faltan por dar trámite
- ✓ Para Saber cuál es el Histórico de un Determinado documento
- ✓ Para Identificar el cuello de Botella en los trámites de Documentos

Se pretende optimizar el desempeño del departamento legal de la entidad financiera donde de implementara el Sistema Visado de Poderes, mediante el adecuado sistema de distribución de carga laboral esto se lograra implementando el algoritmo de banquero

1.4.1 Alcances del estudio

Desarrollar el sistema Visado de Poderes aplicando el **Algoritmo del Banquero** para la distribución de carga de trabajo para el departamento legal de la entidad financiera donde se implantara dicho sistema.

1.5 Propuesta

Se propone la implementación del Sistema Visado de Poderes el cual aplicara el algoritmo del Banquero para la distribución automática de la carga laboral para el departamento de Servicios Jurídicos de la entidad financiera donde se llevara a cabo la implementación del sistema; para este caso la carga laboral serán las distintas solicitudes registradas en el Sistema de Visado de Poderes, a distribuir entre los recursos competentes en el departamento legal, los cuales también se tendrán que registrar en el sistema.

Mediante la implementación del algoritmo Banquero en la distribución de carga automática se lograra una mejor asignación de los recursos competentes.

El proceso luego de la implementación consistirá en subir los documentos a validar al Sistema Visado de Poderes, el cual se encarga de distribuir los documentos entre los recursos de los distintos estudios de abogados del cual dispone el área legal, para su verificación.

A continuación se muestra una tabla estadística con algunos de los servicios brindados en cual nos permite tener una media del tiempo que lleva realizar estos trámites mediante el Sistema Visado de Poderes luego de su implementación.

Tramites	Documentos necesarios por tramite	Tiempo promedio en cargar y distribuir los documentos mediante el sistema	Tiempo promedio en asignar a un trámite a un recurso (abogado) mediante el sistema	Tiempo promedio en retornar los documentos a la oficina de origen	Tiempo promedio empleado por recursos para validar el tramite	Tiempo total promedio empleado para dar una respuesta sobre el tramite realizado en minutos mediante el sistema
REGISTRO EN EL SISTEMA DE APODERADO	✓ COPIA DNI ✓ ORIGINAL PARTIDA	1 - 2 minutos	1 - 3 segundos	0 minutos	1 - 2 minutos	3 - 7 minutos
CLIENTES FALLECIDOS CON TESTAMENTO	✓ ORIGINAL DE LA PARTIDA ✓ CARTA DE SOLICITUD ✓ COPIA DEL TESTAMENTO ✓ COPIA DNI	2 - 5 minutos	1 - 3 segundos	0 minutos	2 - 5 minutos	5 - 13 minutos
CLIENTES FALLECIDOS SIN TESTAMENTO	✓ COPIA DNI	1 - 2 minutos	1 - 3 segundos	0 minutos	1 - 2 minutos	3 - 7 minutos

Tabla 2 Cuadro estadístico posterior a la implementación

En el cuadro anterior se puede observar la reducción de los tiempos en la distribución de los archivos (Documentos adjuntos), así como el tiempo de asignación de los trámites a los recursos competentes mediante la implementación del algoritmo del banquero.

1.6 Organización de la tesis

La organización de este trabajo estará dividida principalmente en 5 capítulos:

El primer capítulo, Introducción al trabajo la cual estará compuesta por los Antecedentes, Definición del Problema, los Objetivos, la Justificación y la Propuesta.

El segundo capítulo, Marco teórico, se definirán conceptos claves para comprender el proyecto de investigación como el algoritmo del Banquero entre otros.

El tercer capítulo, Estado del Arte donde se describirán las distintas alternativas que permiten dar solución al problema planteado.

El cuarto capítulo, Aporte Teórico abordara la justificación de la técnica usada, se elabora un benchmarking entre las distintas alternativas, así también se describirá la solución planteada y sus principales características.

El quinto capítulo, Aporte Practico este capítulo abordara el desarrollo del trabajo así como la implementación de la técnica usada.

El sexto y último capítulo, Conclusiones y Sugerencias se mencionaran las principales conclusiones a la que se llegaron durante la elaboración del proyecto desarrollado además se harán recomendaciones sobre el trabajo y la técnica usada; también se mencionaran posibles futuros trabajos a realizar.

CAPÍTULO II. MARCO TEÓRICO

Se mostrara la ubicación del tema en su contexto y se dará las definiciones que nos permitirán entender con mayor claridad el presente trabajo.

2.1 Sistemas de tramite Documentario

El objetivo principal del Sistema de Trámite Documentario es permitir a las Organizaciones tener el control de la ubicación física y lógica de la documentación que llega y fluye dentro de ella, así como de la que se genera al interior de la misma.[15]

Figura 1 Sistema de Trámite Documentario fuente [14]

2.2 Ciclo vital del documento

En el ciclo vital del documento cabe destacar que encontramos los siguientes conceptos:

- ✓ producción documental.
- ✓ recepción de documentos.

- ✓ distribución de documentos.
- ✓ trámite de documentos.
- ✓ organización de documentos.
- ✓ consulta de documentos.
- ✓ conservación de documentos.
- ✓ disposición final de documentos.

[14]

De los conceptos mencionados anteriormente cabe resaltar dos conceptos importantes dentro de un sistema de trámite documentario como son la distribución y consulta de documentos, El primero debe dar garantía que el documento llegue a su destinatario y el segundo de permitir el acceso a un documento o grupo de documentos con el fin de conocer la información que contienen.

Figura 2 Distribución y Consulta de documentos fuente [14]

2.3 Abrazo mortal

En sistemas operativos, el bloqueo mutuo (también conocido como interbloqueo, traba mortal, deadlock, abrazo mortal) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. A diferencia de otros problemas de concurrencia de 7procesos, no existe una solución general para los interbloqueos.

Todos los interbloqueos surgen de necesidades que no pueden ser satisfechas, por parte de dos o más procesos. En la vida real, un ejemplo puede ser el de dos niños que intentan jugar al arco y flecha, uno toma el arco, el otro la flecha. Ninguno puede jugar hasta que alguno libere lo que tomó.

En el siguiente ejemplo, dos procesos compiten por dos recursos que necesitan para funcionar, que sólo pueden ser utilizados por un proceso a la vez. El primer proceso obtiene el permiso de utilizar uno de los recursos (adquiere el lock sobre ese recurso). El segundo proceso toma el lock del otro recurso, y luego intenta utilizar el recurso ya utilizado por el primer proceso, por lo tanto queda en espera. Cuando el primer proceso a su vez intenta utilizar el otro recurso, se produce un interbloqueo, donde los dos procesos esperan la liberación del recurso que utiliza el otro proceso. [9]

2.3.1 Ejemplos de Interbloqueo

Ejemplo 1: Interbloqueo de tráfico

Cuatro coches llegan aproximadamente en el mismo instante a un cruce de cuatro caminos. Los cuatro cuadrantes de la intersección son los recursos compartidos sobre los que se demanda control; por tanto, si los coches desean atravesar el cruce, las necesidades de recursos son las siguientes:

- - El coche que va hacia el norte necesita los cuadrantes 1 y 2.
- - El coche que va hacia el oeste necesita los cuadrantes 2 y 3.
- - El coche que va hacia el sur necesita los cuadrantes 3 y 4.
- - El coche que va hacia el este necesita los cuadrantes 4 y 1.

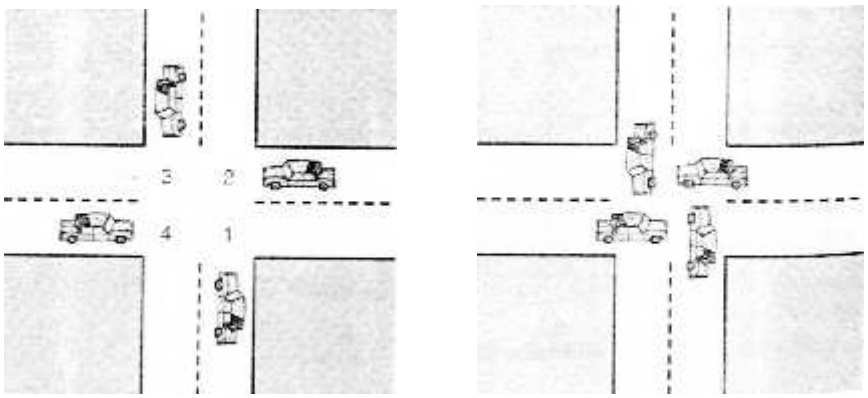


Figura 3 Interbloqueo de tráfico fuente [9]

La norma más habitual en la carretera es que un coche en un cruce de cuatro caminos debe ceder el paso al coche que está a su derecha. Esta norma funciona si solo hay dos o tres coches en el cruce. Por ejemplo, si solo llegan al cruce los coches del norte y del oeste, el coche del norte esperará hasta que el del oeste pase. Sin embargo, si los cuatro coches llegan al mismo tiempo cada uno se abstendrá de entrar en el cruce, provocando interbloqueo. Si todos los coches ignoran las normas y entran (con cuidado) en el cruce, cada coche obtendrá un recurso (un cuadrante) pero no podrá continuar porque el segundo recurso que necesita ya ha sido invadido por otro coche. De nuevo, se tiene interbloqueo.

Ejemplo 2: Cruce en un puente (es parecido al interbloqueo de tráfico)

En una carretera de dos direcciones, donde en un determinado cruce con la vía del ferrocarril, se ha construido un puente que solo deja pasar vehículos en un solo sentido. El bloqueo ocurre cuando dos carros intentan pasar por el puente al mismo tiempo.

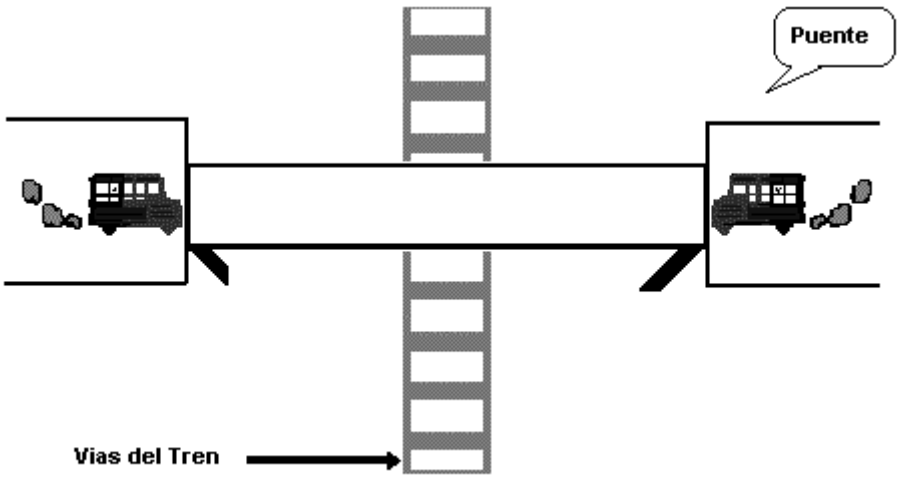


Figura 4 Cruce en un puente fuente [9]

Una manera de resolver el bloqueo es: el conductor situado en uno de los extremos es lo suficientemente educado que deja pasar en primer lugar al del otro extremo y luego pasa él.

Este ejemplo nos muestra como sucede el interbloqueo en nuestra vida diaria.

Ejemplo 3 Procesos

Dos procesos desean imprimir cada uno un enorme archivo en cinta. El proceso **A** solicita el permiso para utilizar la impresora, el cual se le concede. Es entonces cuando el proceso **B** solicita permiso para utilizar la unidad de cinta y se le otorga. El proceso **A** solicita entonces la unidad de cinta, pero la solicitud es denegada hasta que **B** la libere. Por desgracia, en este momento, en vez de liberar unidad de cinta, **B** solicita la impresora. Los procesos se bloquean en ese momento y permanecen así por siempre. [16]

2.3.2 Representación de Bloqueos Mutuos usando grafos

El Bloqueo mutuo también puede ser representado usando grafos dirigidos, donde el proceso es representado por un círculo y el recurso, por un cuadrado. Cuando un

proceso solicita un recurso, una flecha es dirigida del círculo al cuadrado. Cuando un recurso es asignado a un proceso, una flecha es dirigida del cuadrado al círculo.

En la figura del ejemplo, se pueden ver dos procesos diferentes (A y B), cada uno con un recurso diferente asignado (R1 y R2). En este ejemplo clásico de bloqueo mutuo, es fácilmente visible la condición de espera circular en la que los procesos se encuentran, donde cada uno solicita un recurso que está asignado a otro proceso. [9]

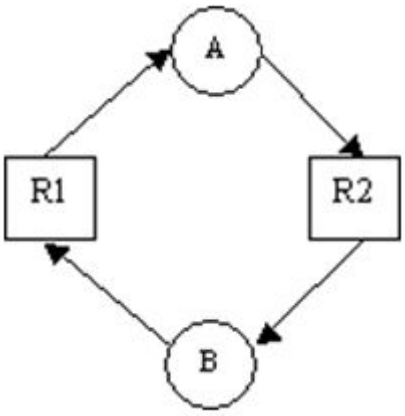


Figura 5 Abrazo mortal fuente [9]

2.3.3 Modelación de Bloqueos

La *modelación de bloqueos* se puede mostrar mediante gráficas dirigidas

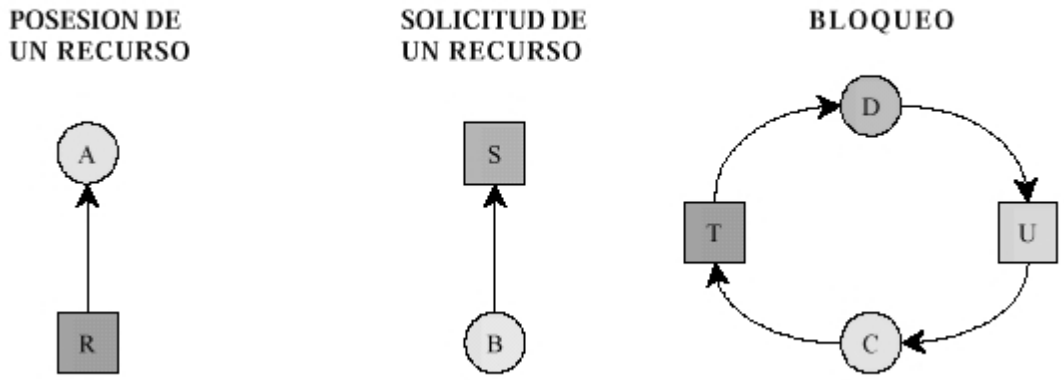


Figura 6 Graficas de asignación de recursos fuente [1]

Las gráficas tienen dos tipos de nodos:

- Procesos (aparecen como círculos).
- Recursos (aparecen como cuadrados).
- Un arco de un nodo de recurso a uno de proceso indica que el recurso fue solicitado con anterioridad, fue otorgado y es poseído en ese momento por dicho proceso.
- Un arco de un proceso a un recurso indica que el proceso está bloqueado, en espera de ese recurso.
- Un ciclo en la gráfica indica la *existencia de un bloqueo* relacionado con los procesos y recursos en el ciclo.

PROCESOS: A, B, C RECURSOS: R, S, T

SECUENCIA DEL PROCESO A:
SOLICITUD DE R, SOLICITUD DE S, LIBERACION DE R, LIBERACION DE S.

SECUENCIA DEL PROCESO B:
SOLICITUD DE S, SOLICITUD DE T, LIBERACION DE S, LIBERACION DE T.

SECUENCIA DEL PROCESO C:
SOLICITUD DE T, SOLICITUD DE R, LIBERACION DE T, LIBERACION DE R.

SECUENCIA DE SOLICITUDES DE RECURSOS QUE CONDUCE A BLOQUEO:

A SOLICITUD R, B SOLICITUD S, C SOLICITUD T,
A SOLICITUD S, B SOLICITUD T, C SOLICITUD R, BLOQUEO.

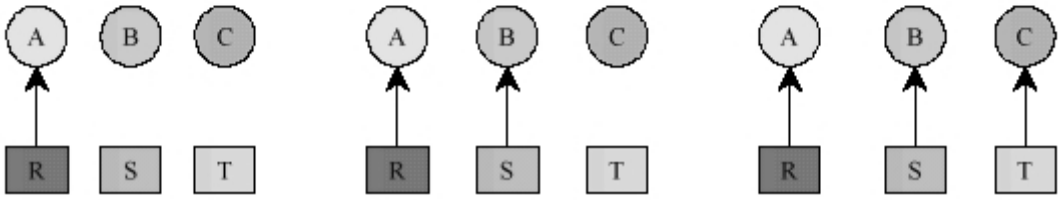


Figura 7 Ocurrencia de un bloqueo y forma de evitarlo fuente [1]

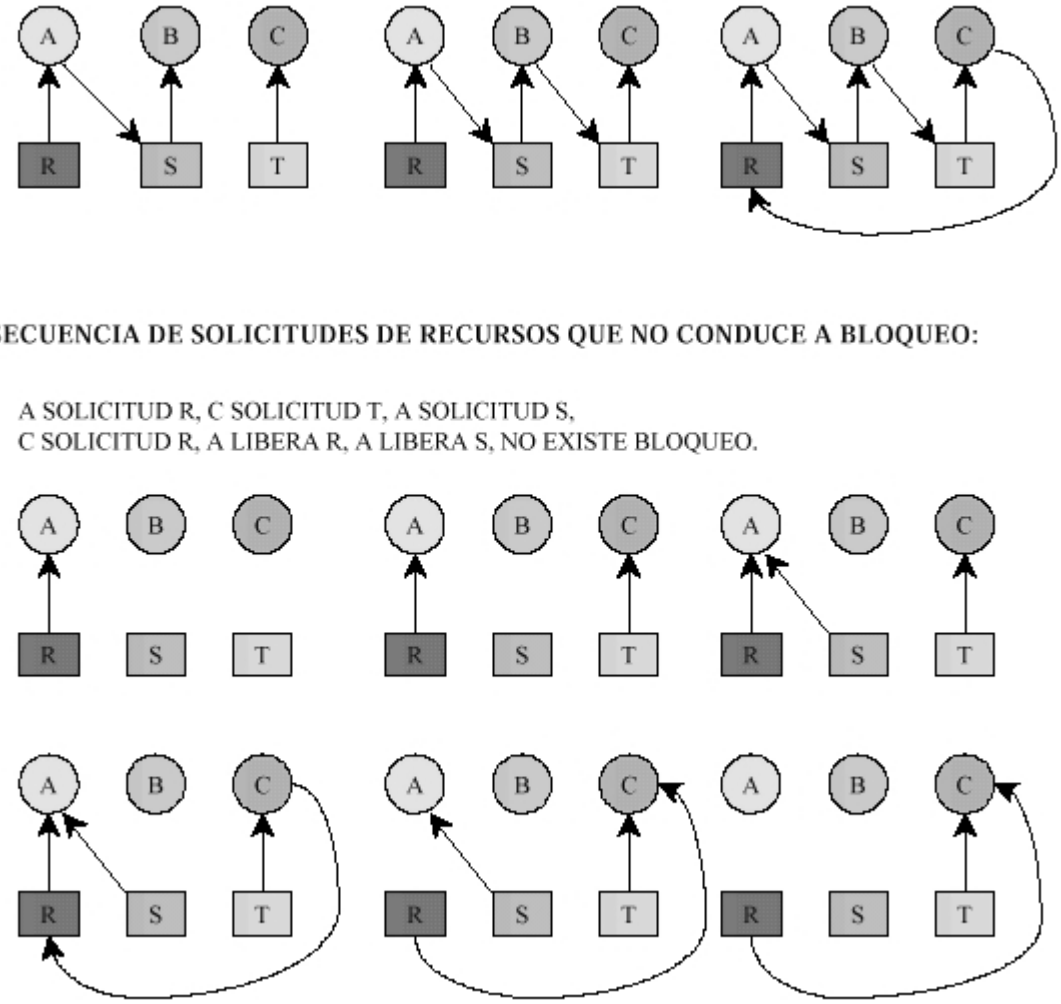


Figura 8 Ocurrencia de un bloqueo y forma de evitarlo continuación fuente [1]

[1]

2.3.4 Condiciones necesarias

También conocidas como condiciones de Coffman por su primera descripción en 1971 en un artículo escrito por E. G. Coffman.

Estas condiciones deben cumplirse simultáneamente y no son totalmente independientes entre ellas.

Sean los procesos P0, P1,..., Pn y los recursos R0, R1,..., Rm:

- Condición de exclusión mutua: Existencia al menos de un recurso compartido por los procesos, al cual sólo puede acceder uno simultáneamente.
- Condición de posesión y espera: Al menos un proceso P_i ha adquirido un recurso R_i, y lo mantiene mientras espera al menos un recurso *Rj* que ya ha sido asignado a otro proceso.
- Condición de no expropiación: Los recursos no pueden ser apropiados por los procesos, es decir, los recursos sólo podrán ser liberados voluntariamente por sus propietarios.
- Condición de espera circular: Dado el conjunto de procesos $P_0...P_n$, P_0 está esperando un recurso adquirido por P_1, que está esperando un recurso adquirido por P_2,... que está esperando un recurso adquirido por P_n, que está esperando un recurso adquirido por P_0. Esta condición implica la condición de retención y espera. [9][3][4]

2.3.5 Evitando bloqueos mutuos

Los bloqueos mutuos pueden ser evitados si se sabe cierta información sobre los procesos antes de la asignación de recursos. Para cada petición de recursos, el sistema controla si satisfaciendo el pedido entra en un estado inseguro, donde puede producirse un bloqueo mutuo. De esta forma, el sistema satisface los pedidos de recursos solamente si se asegura que quedará en un estado seguro. Para que el sistema sea capaz de decidir si el siguiente estado será seguro o inseguro, debe saber por adelantado y en cualquier momento el número y tipo de todos los recursos en existencia, disponibles y requeridos. Existen varios algoritmos para evitar bloqueos mutuos:

- ✓ Algoritmo del banquero, introducido por Dijkstra.
- ✓ Algoritmo de grafo de asignación de recursos.
- ✓ Algoritmo de Seguridad.
- ✓ Algoritmo de solicitud de recursos.

[9]

2.3.6 Prevención

Los bloqueos mutuos pueden prevenirse asegurando que no suceda alguna de las condiciones necesarias vistas anteriormente.

- Eliminando la exclusión mutua: ningún proceso puede tener acceso exclusivo a un recurso. Esto es imposible para procesos que no pueden ser encolados (puestos en un spool), e incluso con colas también pueden ocurrir interbloqueos.
- La condición de posesión y espera puede ser eliminada haciendo que los procesos pidan todos los recursos que van a necesitar antes de empezar. Este conocimiento por adelantado muchas veces es imposible nuevamente. Otra forma es requerir a los procesos liberar todos sus recursos antes de pedir todos los recursos que necesitan. Esto también es poco práctico en general.
- La condición de no expropiación puede ser también imposible de eliminar dado que un proceso debe poder tener un recurso por un cierto tiempo o el procesamiento puede quedar inconsistente.
- La condición de espera circular es la más fácil de atacar. Se le permite a un proceso poseer sólo un recurso en un determinado momento, o una jerarquía puede ser impuesta de modo tal que los ciclos de espera no sean posibles.[9]

2.4 Estado seguro

Un sistema se encuentra en un estado seguro si existe un orden en que pueden concederse las peticiones de recursos a todos los procesos, previniendo el interbloqueo. El algoritmo del banquero funciona encontrando estados de este tipo.

Los procesos piden recursos, y son complacidos siempre y cuando el sistema se mantenga en un estado seguro después de la concesión. De lo contrario, el proceso es suspendido hasta que otro proceso libere recursos suficientes.

En términos más formales, un sistema se encuentra en un estado seguro si existe una secuencia segura. Una secuencia segura es una sucesión de procesos, < P1,..., Pn >, donde para un proceso Pi, el pedido de recursos puede ser satisfecho con los recursos disponibles sumados los recursos que están siendo utilizados por Pj, donde j < i. Si no

hay suficientes recursos para el proceso Pi, debe esperar hasta que algún proceso Pj termine su ejecución y libere sus recursos. Recién entonces podrá Pi tomar los recursos necesarios, utilizarlos y terminar su ejecución. Al suceder esto, el proceso Pi+1 puede tomar los recursos que necesite, y así sucesivamente. Si una secuencia de este tipo no existe, el sistema se dice que está en un estado inseguro, aunque esto no implica que esté bloqueado. [17]

2.4.1 Ejemplos de Estado

Ejemplo de Estado seguro

Supóngase que un sistema tiene doce unidades de cinta y tres procesos que las comparten.

	Préstamo actual	Necesidad máxima
Proceso 1	1	4
Proceso 2	4	6
Proceso 3	5	8
Disponibles	2	

Figura 9 Estado seguro fuente [17]

La tabla anterior representa un estado seguro porque el proceso 2 tiene un préstamo de 4 unidades y necesita como máximo 6, o sea, 2 más. El sistema tiene 12 de las cuales 10 están en uso y mantiene 2 disponibles. Si las que están disponibles se asignan al proceso 2, cubriendo su demanda máxima, este proceso podrá terminar. Al acabar, devolverá todos los recursos, 6 unidades de cinta, y el sistema podrá asignarlas al proceso 1 y al 3. De esta forma, la clave de un estado seguro es que exista al menos una forma en la que terminen todos los procesos.

Ejemplo de Estado inseguro

	Préstamo actual	Necesidad máxima
Proceso 1	8	10
Proceso 2	2	5
Proceso 3	1	3
Disponibles	1	

Figura 10 Estado inseguro fuente [17]

Ahora 11 de las 12 unidades de cinta están asignadas y solamente hay una disponible. En este momento, no se puede garantizar que terminen los tres procesos. Si el proceso 1 pide y obtiene la última unidad de cinta y los tres vuelven a solicitar una unidad de cinta más se produciría un bloqueo triple.

Es importante señalar, que un estado inseguro no implica la existencia, ni siquiera eventual, de un interbloqueo. Lo que sí implica un estado inseguro es la posibilidad de que ocurra por una desafortunada secuencia de eventos.

Ejemplo de transición de Estado seguro a Estado inseguro

Saber que un estado es seguro no implica que serán seguros todos los estados futuros. La política de asignación de recursos debe considerar cuidadosamente todas las peticiones antes de satisfacerlas. Por ejemplo supongamos la situación que se muestra en la siguiente tabla.

	Préstamo actual	Necesidad máxima
Proceso 1	1	4
Proceso 2	4	6
Proceso 3	5	8
Disponibles	2	

Figura 11 Transición de Estado seguro a Estado inseguro fuente [17]

Ahora supóngase que el proceso 3 pide un recurso más. Si el sistema satisface esta petición, el nuevo estado será el que se muestra en la tabla de abajo.

	Préstamo actual	Necesidad máxima
Proceso 1	1	4
Proceso 2	4	6
Proceso 3	5	8
Disponibles	1	

Figura 12 Transición de estado seguro a Estado inseguro fuente [17]

Según vemos, aunque, en principio el sistema no estaba bloqueado, ha pasado de un estado seguro a uno inseguro. La última de las tablas caracteriza un sistema en el cual no puede garantizarse la terminación de todos los procesos. Solamente hay un recurso disponible, pero deben estarlo al menos dos para asegurar que el proceso 2 o el 3 puedan terminar, devolver sus recursos al sistema y permitir que los otros procesos acaben. [17]

2.5 Definición algoritmo Banquero

El Algoritmo del banquero, en sistemas operativos es una forma de evitar el interbloqueo, propuesta por primera vez por Edsger Dijkstra. Es un acercamiento teórico para evitar los interbloqueos en la planificación de recursos. Requiere conocer con anticipación los recursos que serán utilizados por todos los procesos. Esto último generalmente no puede ser satisfecho en la práctica.

Este algoritmo usualmente es explicado usando la analogía con el funcionamiento de un banco. Los clientes representan a los procesos, que tienen un crédito límite, y el dinero representa a los recursos. El banquero es el sistema operativo.

El banco confía en que no tendrá que permitir a todos sus clientes la utilización de todo su crédito a la vez. El banco también asume que si un cliente maximiza su crédito será capaz de terminar sus negocios y devolver el dinero a la entidad, permitiendo servir a otros clientes.

El algoritmo mantiene al sistema en un **estado seguro**. Un sistema se encuentra en un estado seguro si existe un orden en que pueden concederse las peticiones de recursos a todos los procesos, previniendo el interbloqueo. El algoritmo del banquero funciona encontrando estados de este tipo.

Los procesos piden recursos, y son complacidos siempre y cuando el sistema se mantenga en un estado seguro después de la concesión. De lo contrario, el proceso es suspendido hasta que otro proceso libere recursos suficientes.

En términos más formales, un sistema se encuentra en un estado seguro si existe una secuencia segura. Una secuencia segura es una sucesión de procesos, $< P_1,..., P_n >$, donde para un proceso P_i, el pedido de recursos puede ser satisfecho con los recursos disponibles sumados los recursos que están siendo utilizados por P_j, donde j < i. Si no hay suficientes recursos para el proceso P_i, debe esperar hasta que algún proceso P_j termine su ejecución y libere sus recursos. Recién entonces podrá P_i tomar los recursos necesarios, utilizarlos y terminar su ejecución. Al suceder esto, el proceso P_{i+1} puede tomar los recursos que necesite, y así sucesivamente. Si una secuencia de este tipo no existe, el sistema se dice que está en un **estado inseguro**, aunque esto no implica que esté bloqueado.

Así, el uso de este tipo de algoritmo permite impedir el interbloqueo, pero supone una serie de restricciones:

- Se debe conocer la máxima demanda de recursos por anticipado.
- Los procesos deben ser independientes, es decir que puedan ser ejecutados en cualquier orden. Por lo tanto su ejecución no debe estar forzada por condiciones de sincronización.
- Debe haber un número fijo de recursos a utilizar y un número fijo de procesos.
- Los procesos no pueden finalizar mientras retengan recursos.

[6]

2.5.1 Estructuras y complejidad

Se utilizan cuatro vectores: recursos, asignación, demanda y disponible. La información que guarda cada uno es la siguiente:

- Recursos: este vector mantiene la cantidad total de recursos que pueden ser utilizados por los procesos. De esta forma, Recursos[i] = k significa que hay una cantidad total k de recursos R_i disponibles.
- Asignación: en esta matriz constan la actual asignación de recursos a cada uno de los procesos. Asignación[i][j] = k significa que el proceso número i tiene asignado k unidades del recurso j.
- Demanda: esta matriz guarda las cantidades máximas de recursos de cada tipo que serán utilizados por cada proceso.
- Disponible: en cualquier momento, el vector Disponible guardará la cantidad disponible de cada recurso. Disponible[i] = k significa que hay una cantidad de k recursos i disponibles para ser utilizados.

En términos de complejidad, el algoritmo del banquero es de orden $O(n^2 \times m)$, donde *n* es el número de procesos y *m* la cantidad de recursos. [6]

La complejidad de los algoritmos nos representa o dice el tiempo de ejecución de cualquier programa en base a los 'n' datos de entrada.

2.6 UML

Lenguaje Unificado de Modelado (LUM o **UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

[13]

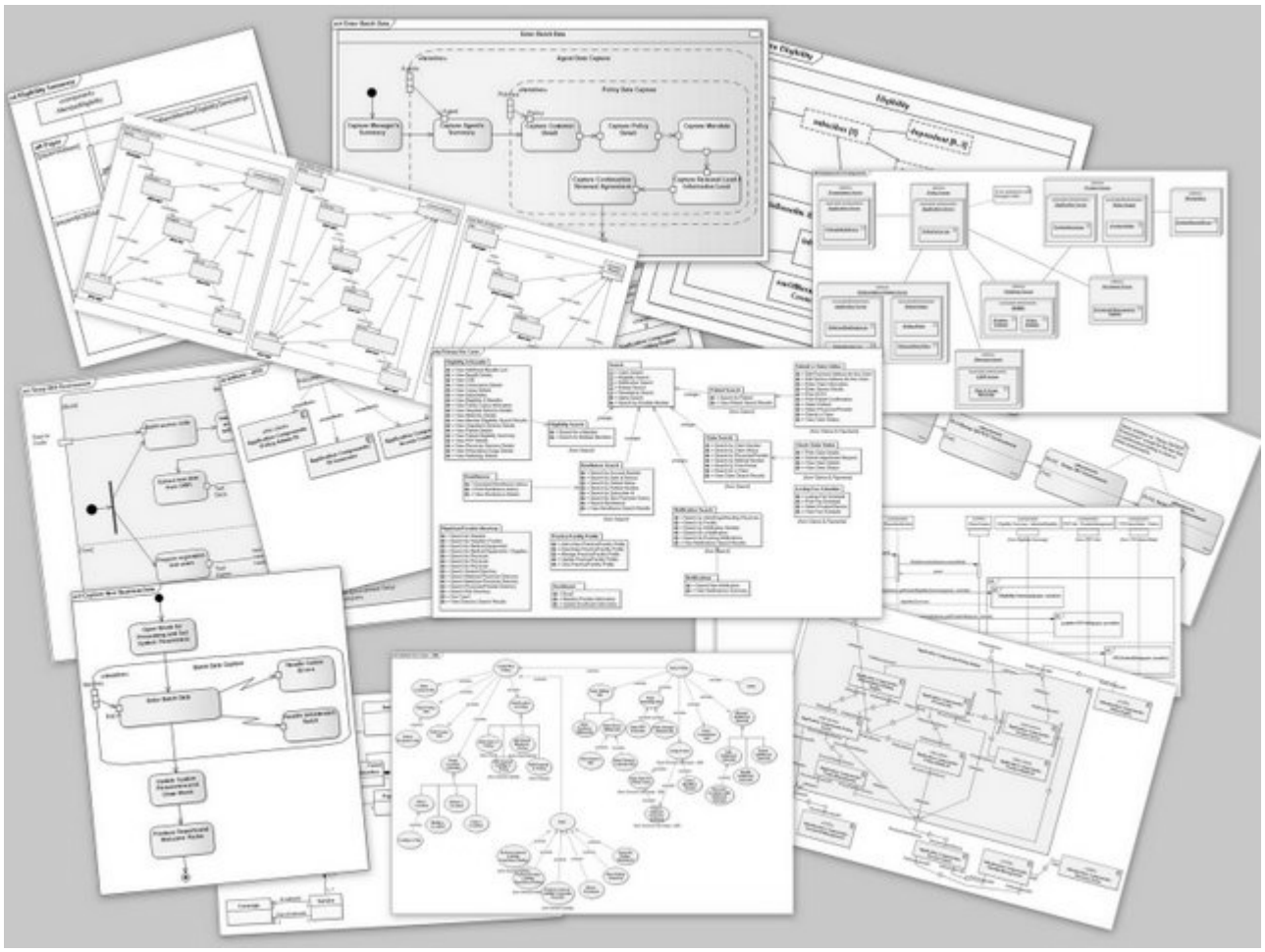


Figura 13 UML diagramas fuente [13]

2.6.1 Diagramas

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente, como se muestra en la figura de la derecha.

Los ***Diagramas de Estructura*** enfatizan en los elementos que deben existir en el sistema modelado:

- ✓ Diagrama de clases
- ✓ Diagrama de componentes
- ✓ Diagrama de objetos
- ✓ Diagrama de estructura compuesta (UML 2.0)

- ✓ Diagrama de despliegue
- ✓ Diagrama de paquetes

Los ***Diagramas de Comportamiento*** enfatizan en lo que debe suceder en el sistema modelado:

- ✓ Diagrama de actividades
- ✓ Diagrama de casos de uso
- ✓ Diagrama de estados

Los ***Diagramas de Interacción*** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- ✓ Diagrama de secuencia
- ✓ Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x)
- ✓ Diagrama de tiempos (UML 2.0)
- ✓ Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

[13]

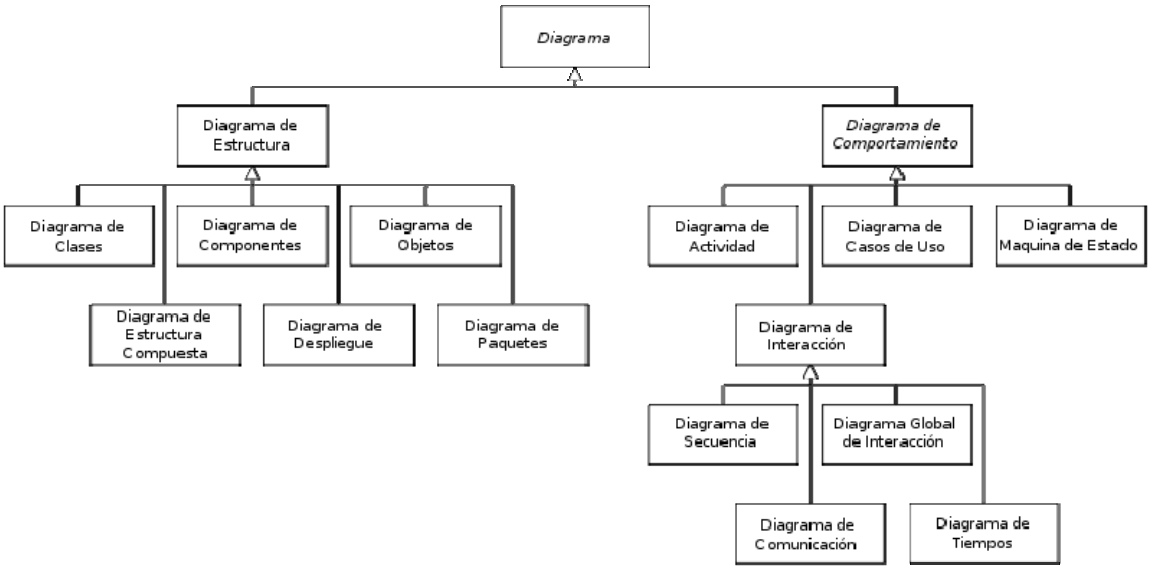


Figura 14 Jerarquía de Diagramas fuente [13]

2.7 JAVA

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

[12]

CAPÍTULO III. ESTADO DEL ARTE METODOLÓGICO

3.1 Taxonomía

Las ciencias de la computación son aquellas que abarcan el estudio de las bases teóricas de la información y la computación, así como su aplicación en sistemas computacionales. Existen diversos campos o disciplinas dentro de las Ciencias de la Computación o Ciencias Computacionales; algunos enfatizan los resultados específicos del cómputo (como los gráficos por computadora), mientras que otros (como la teoría de la complejidad computacional) se relacionan con propiedades de los algoritmos usados al realizar cómputos. Otros por su parte se enfocan en los problemas que requieren la implementación de cómputos. Por ejemplo, los estudios de la teoría de lenguajes de programación describen un cómputo, mientras que la programación de computadoras aplica lenguajes de programación específicos para desarrollar una solución a un problema computacional concreto. La informática se refiere al tratamiento automatizado de la información de una forma útil y oportuna. No se debe confundir el carácter teórico de esta ciencia con otros aspectos prácticos como Internet.

De acuerdo a Peter J. Denning, la cuestión fundamental en que se basa la ciencia de la computación es, "¿Qué puede ser (eficientemente) automatizado?".

Campos de las ciencias de la computación :

- ✓ Fundamentos matemáticos
- ✓ Teoría de la computación
- ✓ Algoritmos y estructuras de datos
- ✓ Lenguajes de programación y compiladores
- ✓ Bases de datos
- ✓ Sistemas concurrentes, paralelos y distribuidos
- ✓ Inteligencia artificial
- ✓ Gráficos por computador

- ✓ Computación científica

[10]

El algoritmo del banquero se encuentra dentro del campo de estudio de algoritmos y estructuras de datos de las ciencias de la computación; este algoritmo es usado para la asignación de recursos o tratamiento de interbloqueos por los sistemas operativos

Existen métodos y algoritmos para asignación de recursos en otras áreas de estudio como por ejemplo en el Área de estudio de la investigación operativa con el método de la programación lineal.

Nos centraremos principalmente en el Área de estudio de los sistemas Operativos en los tipos de algoritmos de Asignación de recursos o Algoritmos para evitar interbloqueos.

3.1.1 Sistemas operativos: Algoritmo del banquero

Se le nombro así al algoritmo porque puede usarse en un sistema bancario para asegurar que el banquero controle la asignación de efectivo disponible para satisfacer las necesidades de todos sus clientes evitando quedar sin efectivo.

3.1.1.1 Explicación del algoritmo:

Sea n el número de procesos del sistema, y m el número de tipos de recursos. Necesitamos las siguientes estructuras de datos:

Disponible: Un vector de longitud m indica el numero de recursos disponibles de cada tipo. Si Disponible[i] = k, hay k ejemplares disponibles del tipo de recursos Rj

Max: Una matriz n x m define la demanda máxima de cada proceso, si Max[i, j] = k el proceso Pi puede solicitar más k ejemplares del tipo de recursos Rj

Asignación: Una matriz n x m define el número de procesos, si Asignacion[i, j] = k, el proceso Pi tiene asignados actualmente k ejemplares del tipo de recursos Rj

Necesidad: Una matriz n x m indica los recursos que todavía le hacen falta a cada proceso, si Necesidad[i, j] = k, el proceso Pi podría necesitar k ejemplares mas del tipo de recursos Rj para llevar a cabo su tarea. Observe que Necesidad[i, j] = Max[i, j] - Asignacion[i, j].

3.1.1.2 Algoritmo de seguridad:

Utilizado para averiguar si un sistema está o no en un estado seguro, se puede describir como sigue:

Sean Trabajo y Fin vectores con longitud m y n respectivamente, Asignar los valores iníciales Trabajo := Disponible y Fin[i] := false para i = 1, 2, ..., n.

Buscar una i tal que:

Fin[i] = false, y

Necesidadi <= Trabajo

Si no existe tal i, continuar con el paso 4.

Trabajo:=Trabajo+Asignacióni
Fin[i]:=true
ir al paso 2.

Si Fin[i] = true para toda i, el sistema está en un estado seguro.

Es importante tomar en cuenta que este algoritmo podria requerir m x n2 operaciones para decidir si un estado es seguro o no.

3.1.1.3 Algoritmo de solicitud de recursos

Sea Solicitudi el vector de solicitudes de procesos Pi si Solicitudi[j] = k, el proceso Pi quiere k ejemplares del tipo de recurso Rj Cuando Pj solicita recursos, se emprenden las acciones siguientes:

Si Solicitudi <= Necesidadi ir al paso 2. En caso contrario, indicar una condición de error, pues el proceso ha excedido su reserva máxima.

Si Solicitudi <= Disponible ir al paso 3. En caso contrario Pi Deberá esperar, ya que los recursos no están disponibles.

Hacer que el sistema simule haber asignado al proceso Pi los recursos que solicitó modificando el estado como sigue:

Disponible:=Disponible-Solicitudji

Asignacionj:=Asignacioni+Solicitudji
Nececidadj:=Necesidadj-Solicitudji

El estado de asignación de recursos resultante es seguro, la transacción se llevará a cabo y se asignaran los recursos al proceso Pj pero si el nuevo estado es inseguro, Pj tendrá que esperar Solicitudj y se restaurará el antiguo estado de asignación de recursos.

3.2 Métodos / Modelos / Algoritmos (herramienta teórica)

Dentro del área de estudio de los sistemas operativos existen algoritmos de asignación de recursos o algoritmos para tratamiento de interbloqueos como se le conoce, mencionaremos lo más relevantes así también lo compararemos con el algoritmo seleccionado.

Las estrategias utilizadas para enfrentar los bloqueos son:

- ✓ Ignorar todo el problema.
- ✓ Detección y recuperación.
- ✓ Evitarlos dinámicamente mediante una cuidadosa asignación de recursos.
- ✓ Prevención mediante la negación estructural de una de las cuatro condiciones necesarias.

Veremos a continuación que se puede adoptar alguna estrategia adecuada que nos permitirá prevenir, evitar o detectar y recuperar situaciones de interbloqueo.

3.2.1 Algoritmo de la Avestruz

La estrategia más sencilla es el algoritmo del avestruz: esconder la cabeza bajo tierra y pretender que el problema no existe. La gente reacciona a esta estrategia de distintos modos según su formación. Los matemáticos consideran que es inaceptable y argumentan que los interbloqueos se deben evitar a toda costa. Los ingenieros se interrogan sobre la frecuencia del problema, la frecuencia con el que el sistema se para por otras causas y la importancia de los interbloqueos. Si éstos se presentan de una vez

cada cinco años, y los sistemas se paran una vez al mes por errores en el **hardware**, en el **compilador** o en el sistema operativo, a casi ningún ingeniero le gustaría tener que sufrir una degradación seria de las prestaciones del sistema para garantizar la eliminación de los interbloqueos.

Por ejemplo, Unix pude sufrir interbloqueos que ni siquiera se detectan, y que, por supuesto, no se eliminan automáticamente. El número total de procesos en el sistema viene determinado por el número de posiciones de la tabla de procesos, que, en definitiva, constituye un recurso limitado. Supongamos ahora que un sistema Unix con 100 posiciones en la tabla de procesos tiene ejecutándose diez programas, cada uno de los cuales ha de crear 12 subprocesos. Después de que cada proceso haya creado otros 9, los 10 procesos originales y los 90 nuevos llenarán por completo la tabla. Los 10 procesos originales se encontrarán ahora en un bucle infinito intentando crear un nuevo proceso sin poder: se ha producido un interbloqueo. Otros ejemplos de recursos que suelen ser limitados son: el número máximo de ficheros que pueden estar abiertos está limitado, el área en el disco para ***intercambio con memoria principal***. En realidad, casi todas las tablas del sistema operativo representan recursos limitados, ¿deberíamos, por tanto, limitar estos recursos para no producir un interbloqueo?

La estrategia UNIX es simplemente desentenderse del problema, suponiendo que la mayoría de los usuarios preferirán un interbloqueo ocasional antes que la imposición de que cada usuario pueda crear un solo proceso, abrir un solo fichero y usar sólo una unidad de lo que sea. [17]

3.2.2 Estrategias de Havender

Havender llegó a la conclusión de que si falta alguna de las ***cuatro condiciones necesarias*** no puede haber un interbloqueo. Este autor sugiere las siguientes estrategias para negar varias de esas condiciones:

- ✓ Cada proceso deberá pedir todos sus recursos al mismo tiempo y no podrá seguir la ejecución hasta haberlos recibido todos.

- ✓ Si a un proceso que tiene recursos se le niegan los demás, ese proceso deberá liberar sus recursos y, en caso necesario, pedirlos de nuevo junto con los recursos adicionales.
- ✓ Se impondrá un ordenamiento lineal de los tipos de recursos en todos los procesos; es decir, si a un proceso le han sido asignados recursos de un tipo específico, en lo sucesivo sólo podrá pedir aquellos recursos que siguen en el ordenamiento.

Como vemos Havender presenta tres estrategias y no cuatro. Cada una de ellas, está diseñada para negar una de las condiciones necesarias. La primera de estas condiciones, esto es, que los procesos exijan el uso exclusivo de los recursos que requieren, es una condición que no es deseable impedir, porque específicamente queremos permitir la existencia de ***recursos no compartibles o dedicados***.

Negación de la condición de espera

La primera de las estrategias requiere que los recursos que necesita un proceso sean pedidos de una sola vez. El sistema debe proporcionarlos según el principio de todo o nada. Si está disponible el conjunto de los recursos que necesita un proceso, entonces el sistema puede asignarle todos los recursos y éste seguir su ejecución. Si no está disponible alguno de ellos, el proceso debe esperar. Mientras espera no puede tener ningún recurso. Con esto se elimina la condición de espera y no puede ocurrir un interbloqueo.

Todo esto suena bien, pero puede llevar a un grave desperdicio de recursos. Supongamos que un proceso necesita diez unidades de un determinado recurso para su ejecución. Como debe solicitarlas todas antes de comenzar, los mantendrá en su poder durante toda su ejecución. Pudiera suceder, que el programa únicamente utilice estos recursos al principio de su ejecución, por tanto, los recursos están ociosos el resto del tiempo.

Dividir el programa en varios pasos que se ejecuten de manera relativamente independiente es una técnica empleada con frecuencia para conseguir una mejor utilización de los recursos en estas circunstancias. La asignación de recursos se controla

por etapas. Esta solución reduce el desperdicio pero implica mucho trabajo extra tanto en el diseño de las aplicaciones como en la ejecución.

Por otro lado, esta estrategia puede provocar un ***aplazamiento indefinido***, pues los recursos requeridos pueden no estar disponibles todos al tiempo. El sistema podría, entonces, permitir que se fueran acumulando recursos hasta conseguir todos los que necesita un proceso. Pero mientras se acumulan no se pueden asignar a otros procesos y volvemos a infrautilizarlos.

Negación de la condición de no apropiación

La segunda estrategia de Havender consiste en liberar los recursos que un proceso tiene asignados cuando se le niegan peticiones de recursos adicionales. De esta forma, se anula la ***condición de no apropiación***. Los recursos se pueden quitar al proceso que los tiene antes de que termine su ejecución.

En este caso también existe un costo excesivo. Cuando un proceso libera recursos puede perder todo el trabajo realizado hasta ese momento. El costo puede parecer muy alto, pero la pregunta es: con qué frecuencia ha de pagarse ese precio? Si ocurre de tarde en tarde, entonces éste parece ser un buen método para prevenir el interbloqueo. Si, por el contrario, es muy frecuente, entonces el costo es sustancial y sus efectos demasiado perjudiciales (por ejemplo, para procesos de alta prioridad o ***plazo fijo***).

Esta estrategia también adolece de aplazamiento indefinido. Un proceso puede aplazarse continuamente mientras pide y libera muchas veces los mismos recursos. Si esto ocurre, el sistema puede verse obligado a eliminar el proceso para que otros puedan ejecutarse.

Negación de la condición de espera circular

La tercera estrategia de Havender anula la posibilidad de un ***espera circular***. Como todos los recursos tienen una numeración única y como los procesos deben pedir los recursos en un orden lineal ascendente, es imposible que se presente una espera circular. Esta estrategia presenta las siguientes dificultades:

- ✓ Los recursos deben pedirse en un orden ascendente por número de **recursos**. El número de recurso es asignado por la instalación y debe tener un tiempo de vida

largo (meses o años). Si se agregan nuevos tipos de recursos, puede ser necesario reescribir los programas y los sistemas.

- ✓ Lógicamente, cuando se asignan los números de recursos, éstos deben reflejar el orden normal en que los usan la mayoría de las tareas. Pero los procesos que necesiten los recursos en un orden diferente que el previsto por el sistema, los deberán adquirir y conservar, quizá durante tiempo antes de utilizarlos realmente, lo que significa un desperdicio considerable.
- ✓ Una de las metas más importantes de los sistemas operativos actuales es crear ambientes amables con el usuario. Los usuarios deben ser capaces de desarrollar sus aplicaciones sin tener en cuenta molestas restricciones de hardware y software. El ordenamiento lineal impide al usuario escribir sus códigos libremente. [17]

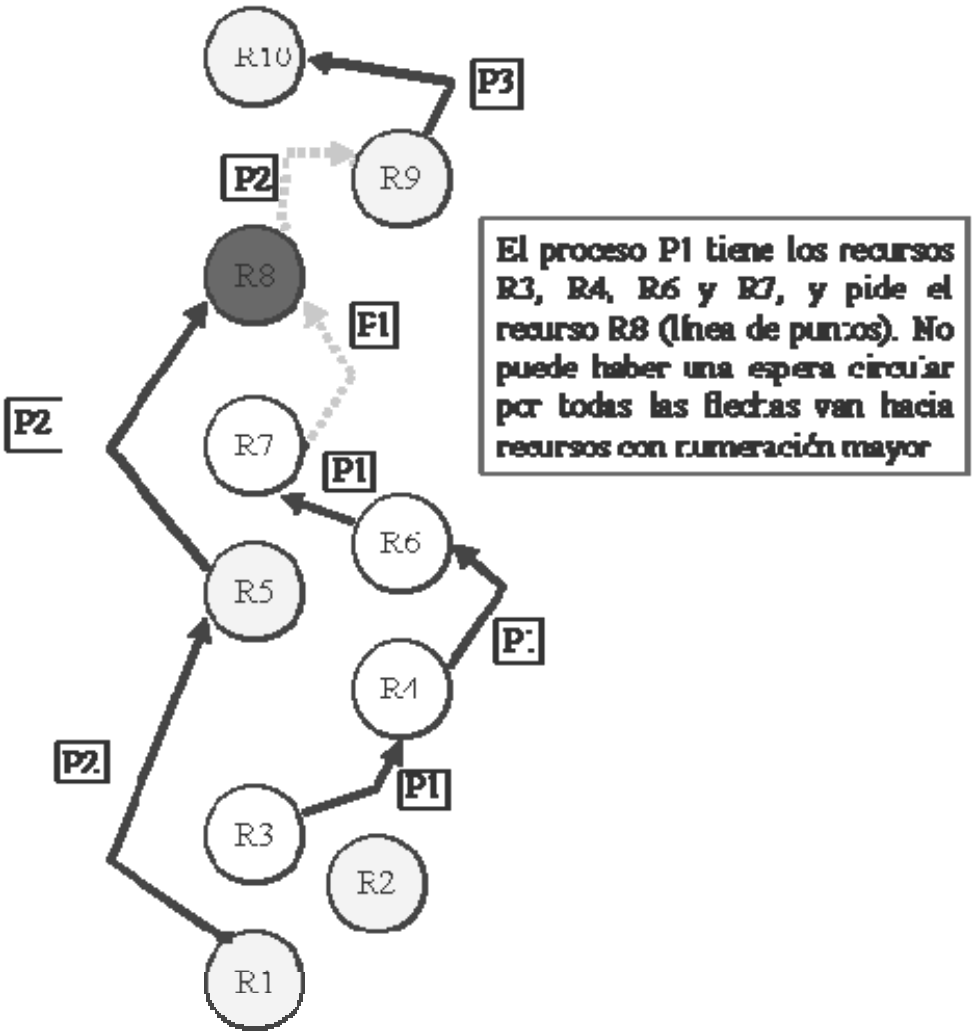


Figura 15 Ordenación lineal de Havender fuente [17]

3.2.3 Reducción de las gráficas de asignación de recursos

Una técnica útil para detectar los interbloqueos consiste en ir reduciendo una gráfica determinando los procesos que pueden completar su ejecución. Si pueden atenderse las peticiones de recursos de un proceso, se dice que la gráfica puede ser reducida por ese proceso. Esta reducción es equivalente a mostrar la gráfica como si el proceso hubiese acabado y hubiera devuelto los recursos al sistema. Si una gráfica puede ser reducida por todos sus procesos, entonces no hay interbloqueo. Si una gráfica no puede ser reducida por todos sus procesos, los procesos irreductibles constituyen el conjunto de procesos en bloqueo mutuo de la gráfica Reducción de graficas.

Cuando se ha bloqueado un sistema, el interbloqueo debe romperse mediante la eliminación de una o más de las condiciones necesarias. Por lo general, varios procesos perderán una parte o la totalidad del trabajo efectuado, pero el precio pagado puede ser pequeño, en comparación con las consecuencias de permitir que el sistema siga bloqueado. La recuperación después de un bloqueo mutuo se complica por varias razones:

- ✓ Puede no estar claro que el sistema se haya bloqueado.
- ✓ La mayor parte de los sistemas tienen medios muy deficientes para suspender indefinidamente un proceso, eliminarlo del sistema y reanudarlo más tarde. De hecho, algunos procesos como los de tiempo real, que deben funcionar continuamente, sencillamente no se pueden suspender y reanudar.
- ✓ Aún cuando existieran medios efectivos de suspensión/reanudación, con toda seguridad implicarían un gasto extra considerable.
- ✓ La recuperación después de un bloqueo mutuo de dimensiones modestas puede significar una cantidad razonable de trabajo, un interbloqueo a gran escala puede requerir una cantidad enorme de trabajo.

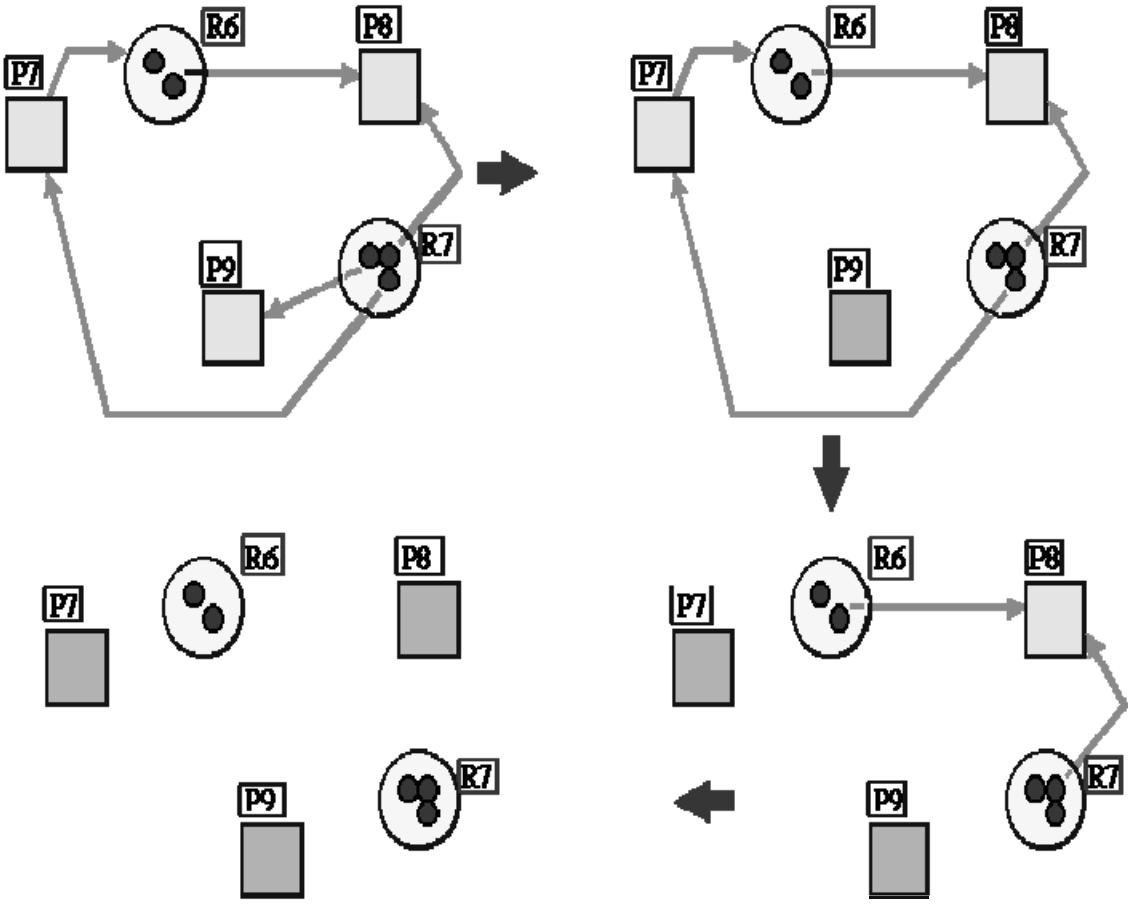


Figura 16 Reducción de graficas fuente [17]

En los sistemas actuales la recuperación se suele realizar eliminado un proceso y arrebatándole sus recursos. Por lo general, el proceso eliminado se pierde pero ahora es posible concluir los procesos restantes. Algunas veces es necesario eliminar varios procesos hasta que se hayan liberado los recursos suficientes para que terminen los procesos restantes.

Los procesos pueden eliminarse de acuerdo con algún orden de **prioridad**. También existen dificultades para ello:

Es posible que no existan prioridades entre los procesos bloqueados, de modo que se tiene que adoptar una decisión arbitraria.

Las prioridades pueden ser incorrectas o un poco confusas debido a consideraciones especiales, como la ***planificación a plazo fijo***, en la cual un proceso de prioridad relativamente baja tiene una prioridad temporal alta a causa de un fin de plazo inminente.

La determinación de una decisión óptima sobre los procesos que se deben eliminar puede requerir un esfuerzo considerable.

Parece ser que el enfoque más deseable para la recuperación después de un bloqueo mutuo sería un mecanismo efectivo de suspensión/reanudación. Ello implicaría suspender temporalmente los procesos y reanudarlos después sin pérdida de trabajo productivo. Para ello sería deseable la posibilidad de especificar **puntos de verificación/reinicio**. De este modo, se facilita la suspensión/reanudación, que se hará desde el último punto de verificación (es decir, la última grabación del estado del sistema). Pero muchos sistemas de aplicación se diseñan sin aprovechar las ventajas de las funciones de punto de verificación/reinicio. Por lo general, se requiere un esfuerzo consciente por parte de los diseñadores para incorporar la verificación/reinicio, y a menos que las tareas requieran muchas horas de ejecución, su uso es poco común. [17]

3.2.4 Benchmarking algoritmos para tratamiento de interbloqueos

Las Estrategias utilizadas para enfrentar los bloqueos por el Dr. **Harvey** M. **Deitel** son:

- ✓ Ignorar todo el problema.
- ✓ Detección y recuperación.
- ✓ Evitarlos dinámicamente mediante una cuidadosa asignación de recursos.
- ✓ Prevención mediante la negación estructural de una de las cuatro condiciones necesarias.

[2]

Benchmarking				
Algoritmos / Estrategias	**Algoritmo del banquero**	**El Algoritmo de la Avestruz**	**Estrategias de Havender**	**Reducción de las gráficas de asignación de recursos**

Asigna recursos	✓	X	✓	X
Previene el interbloqueo	✓X	X	X	X
Evita el interbloqueo	✓	X	X	X
Detecta interbloqueos	✓X	X	X	✓
Recuperación de interbloqueos	X	✓	X	✓
Resultado	**3**	1	1	2

Tabla 3 BenchMarking

Glosario		
X	0	No Aplica
✓	1	Aplica
✓X	1/2	Aplica parcialmente

Tabla 4 Glosario

Como se observa en el resultado de la tabla del benchmarking el algoritmo del banquero cubre más estrategias para el tratamiento de interbloqueos, motivo por el cual se selecciono para su implementación en este proyecto.

3.3 Aplicativos (software)

Su principal aplicación fue en el área de estudio de los sistemas operativos para evitar interbloqueos propuesta por primera vez por Edsger Dijkstra, cuando los recursos del

hardware eran escasos en la actualidad nos encontramos que los recursos ya no son un problema por los cual el algoritmo del banquero dejo de ser aplicado en los sistemas operativo actuales.

Para evitar los interbloqueos en la planificación de recursos. Requiere conocer con anticipación los recursos que serán utilizados por todos los procesos. Esto último generalmente no puede ser satisfecho en la práctica.

En la actualidad el algoritmo de Banquero ya no aplica en los sistemas operativos como se mencione en un párrafo anterior los sistemas operativos prefieren la estrategia desentenderse del problema, suponiendo que la mayoría de los usuarios preferirán un interbloqueo ocasional antes que la imposición de que cada usuario pueda crear un solo proceso, abrir un solo fichero y usar sólo una unidad de lo que sea. Pero este algoritmo es utilizado como técnica de asignación de recursos muy practica.

3.4 Casos de estudio

No se encontró información sobre casos de estudio.

CAPÍTULO IV. DESARROLLO DE LA SOLUCIÓN O DEL ESTUDIO

4.1 Resolución del problema aplicando la técnica seleccionada

El algoritmo será utilizado cada vez que una solicitud pase a estado enviado a servicio jurídico se disparara la distribución automática que utiliza el algoritmo del banquero para que distribuya las solicitudes a los recursos para esto las solicitudes ingresadas serán los procesos y los recursos serán loa abogados

Como se describió en el capitulo anterior se necesitan cuatro vectores de entrada para utilizar el algoritmo del banquero

El primer vector contendrá los recursos, todos los abogados disponibles del área de servicio jurídico todas las instancias de los recursos que para este caso serán n instancias por recurso

El segundo vector contendrá todas las solicitudes ingresadas asignadas a los recursos

El tercer vector la cantidad máxima de recursos por tipo que necesita la solicitud para este caso será de 1 recurso y 1 sola instancia

El cuarto vector contendrá las instancias faltantes por tipo de recuso para este caso también será de 1 recurso y 1 sola instancia

Con las dos últimas condiciones nos aseguramos que siempre el sistema se encuentre en un estado seguro por lo cual se procederá a distribuir las solicitudes sin ningún inconveniente.

4.1.1 Evaluación y selección de la técnica usada

Se selección el Algoritmo del banquero como la técnica más adecuada para asignar recursos debido a las condiciones que presenta el negocio donde se va aplicar el Algoritmo.

Así mismo los parámetro de entrada que utiliza el algoritmo del banquero se pueden condicionar de tal manera que no se necesario modificar el algoritmo y utilizarlo de la manera más conveniente y si en algún momento se desea utilizar el algoritmo en su máxima expresión se envía los parámetros correspondientes siempre y cuando cumpla con las necesidades del negocio.

4.1.1.1 Ventajas del algoritmo del banquero:

- ✓ No es necesario expulsar y hacer retroceder procesos como en la detección del interbloqueo.
- ✓ Es menos restrictivo que la prevención.

4.1.1.2 Desventajas del algoritmo del banquero

El algoritmo del banquero es interesante porque ofrece una forma de asignar los recursos que evita el interbloqueo. Permite ejecutar procesos que tendrían que esperar seguramente con alguna de las estrategias de prevención. Sin embargo, tiene varios defectos importantes:

- ✓ El algoritmo requiere un número fijo de **recursos** asignables. Como los recursos a menudo requieren **servicio**, ya sea por algún fallo o por mantenimiento preventivo, no se puede contar con que será siempre constante.
- ✓ El algoritmo requiere una población de usuarios constantes. En los sistemas ***multiprogramables*** y más en los de tiempo compartido, la población de usuarios cambia constantemente, incluso en cuestión de segundos.
- ✓ El algoritmo requiere que el banquero satisfaga todas las peticiones en un tiempo finito. Es evidente que en los **sistemas reales** esto no es una garantía suficiente.
- ✓ De manera similar, el algoritmo requiere que los procesos salden sus préstamos (es decir, devuelvan sus recursos) en un tiempo finito. Una vez más, esto es insuficiente para un sistema de tiempo real.
- ✓ El algoritmo requiere que los usuarios declaren por anticipado sus necesidades máximas. A medida que la asignación de recursos se hace más dinámica, conocer las necesidades máximas de un usuario presenta mayor dificultad. De hecho, ahora que los sistemas ofrecen **interfaces** gráficas, cada vez es más

común que los usuarios no tengan la menor idea de los recursos que necesitan.[17]

Al parecer existen muchas más desventajas que ventajas pero se selecciono esta técnica debido a que muchas de estas desventajas no son aplicables en el negocio del sistema debido a las condiciones que estas presentan.

4.1.2 Adaptación o aplicación de herramienta teórica para resolver el problema

Se está implementando el algoritmo del banquero bajo un el lenguaje orientado a objetos el cual permite su reutilización y las ventajas que este conlleva, le lenguaje seleccionado es Java

A continuación la implementación de algoritmo en java

```
import java.util.Vector;

public class Banquero {


    prívate  int[] maxRecursos;
    prívate  int numClientes;

    prívate  int[] disponibles;
    prívate  Vector<int[]> max;
    prívate  Vector<int[]> asignados;
    prívate  Vector<int[]> necesidad;


    public Banquero (int ... recursos) {

        maxRecursos = recursos;
        numClientes = 0;

        disponibles = (int[]) recursos.clone();
        max = new Vector();
        asignados = new Vector();
        necesidad = new Vector();
    }

    public int getNumRecursos() {
        return maxRecursos.length;
    }

    public int getNumClientes() {
        return numClientes;
```

```
    }


public void agregarCliente(int[] maxDemanda) throws
IllegalArgumentException {
        if (maxDemanda == null)
      throw new IllegalArgumentException("agregarCliente: La demanda
especificada no puede ser nula en el sistema.");

        if (maxDemanda.length != getNumRecursos())
throw new IllegalArgumentException("agregarCliente: El número de
recursos demandados (" + maxDemanda.length + ") " +"no coincide con el
número de recursos del sistema (" + getNumRecursos() + ").");

        max.addElement(maxDemanda);
        asignados.addElement(new int[maxDemanda.length]);
        necesidad.addElement(maxDemanda.clone());
        numClientes++;
    }


public void cambiarMax(int cid, int[] maxDemanda) throws
IllegalArgumentException, IllegalStateException {
        if (cid < 0 || cid >= getNumClientes())
throw new IllegalArgumentException("cambiarMax: no existe un cliente
con id " + cid + " en el sistema.");

        if (maxDemanda == null)
throw new IllegalArgumentException("cambiarMax: La demanda
especificada no puede ser nula en el sistema.");

        if (maxDemanda.length != getNumRecursos())
throw new IllegalArgumentException("cambiarMax: El número de recursos
demandados (" + maxDemanda.length + ") " +
"no coincide con el número de recursos del sistema (" +
getNumRecursos() + ").");

        int i, asignadosCid[] = asignados.get(cid);
        for (i = 0; i < getNumRecursos(); i++)
            if (asignadosCid[i] != 0)
throw new IllegalStateException("cambiarMax: Solo se puede cambiar la
demanda máxima de un cliente si no tiene ningún recurso asignado.");

        max.set(cid, maxDemanda);
        necesidad.set(cid, maxDemanda.clone());
    }


    public int[] getDisponibles() {
        return disponibles;
    }



public int[] getAsignados(int cid) throws IllegalArgumentException {
        int[] res;
        try {
            res = asignados.get(cid);
        } catch(ArrayIndexOutOfBoundsException aioobe) {
```

```
            res = null;
        }

        if (res == null)
throw new IllegalArgumentException("getAsignados: No existe un cliente
con número " + cid + " en el sistema.");

        return res;
    }



    public int[] getMax(int cid) throws IllegalArgumentException {
        int[] res;
        try {
            res = max.get(cid);
        } catch(ArrayIndexOutOfBoundsException aioobe) {
            res = null;
        }

        if (res == null)
throw new IllegalArgumentException("getMax: No existe un cliente con
número " + cid + " en el sistema.");

        return res;
    }

public int[] getNecesidad(int cid) throws IllegalArgumentException
{
        int[] res;
        try {
            res = necesidad.get(cid);
        } catch(ArrayIndexOutOfBoundsException aioobe) {
            res = null;
        }

        if (res == null)
throw new IllegalArgumentException("getNecesidad: No existe un cliente
con número " + cid + " en el sistema.");

        return res;
    }


    public void otorgaRecursos(int[] recursos, int cid) {
int[] asignadosCid = asignados.get(cid), necesidadCid =
necesidad.get(cid);

        for (int i = 0; i < getNumRecursos(); i++) {
            disponibles[i] -= recursos[i];
            asignadosCid[i] += recursos[i];
            necesidadCid[i] -= recursos[i];
        }
    }



    public void quitaRecursos(int[] recursos, int cid) {
```

```
int[] asignadosCid = asignados.get(cid), necesidadCid =
necesidad.get(cid);

        for (int i = 0; i < getNumRecursos(); i++) {
            disponibles[i] += recursos[i];
            asignadosCid[i] -= recursos[i];
            necesidadCid[i] += recursos[i];
        }
    }


    public void imprimeEstado() {
      int i, j, asignadosCid[], necesidadCid[], numRecursos =
      getNumRecursos();
      System.out.println("-----------------\n Sistema ");
      System.out.println("\nDisponibles:");

for (i = 0; i < numRecursos; System.out.print(String.format("R%1$2d|",
i++)));
        System.out.println();
for (i = 0; i < numRecursos; System.out.print(String.format("%1$3d|",
disponibles[i++])));

        System.out.print("\n\nAsignados:\n   |");
for (j = 0; j < numRecursos; System.out.print(String.format("R%1$2d|",
j++)));

for (i = 0; i < numClientes; i++) {
            asignadosCid = asignados.get(i);
            System.out.print(String.format("\nC%1$2d|", i));

for (j = 0; j < numRecursos; System.out.print(String.format("%1$3d|",
asignadosCid[j++])));
        }

        System.out.print("\n\nNecesidad:\n   |");
for (j = 0; j < numRecursos; System.out.print(String.format("R%1$2d|",
j++)));
            for (i = 0; i < numClientes; i++) {
            necesidadCid = necesidad.get(i);
            System.out.print(String.format("\nC%1$2d|", i));
for (j = 0; j < numRecursos; System.out.print(String.format("%1$3d|",
necesidadCid[j++])));
        }

        System.out.println("\n\n");
    }
}
```

4.2 Descripción de la solución tecnológica

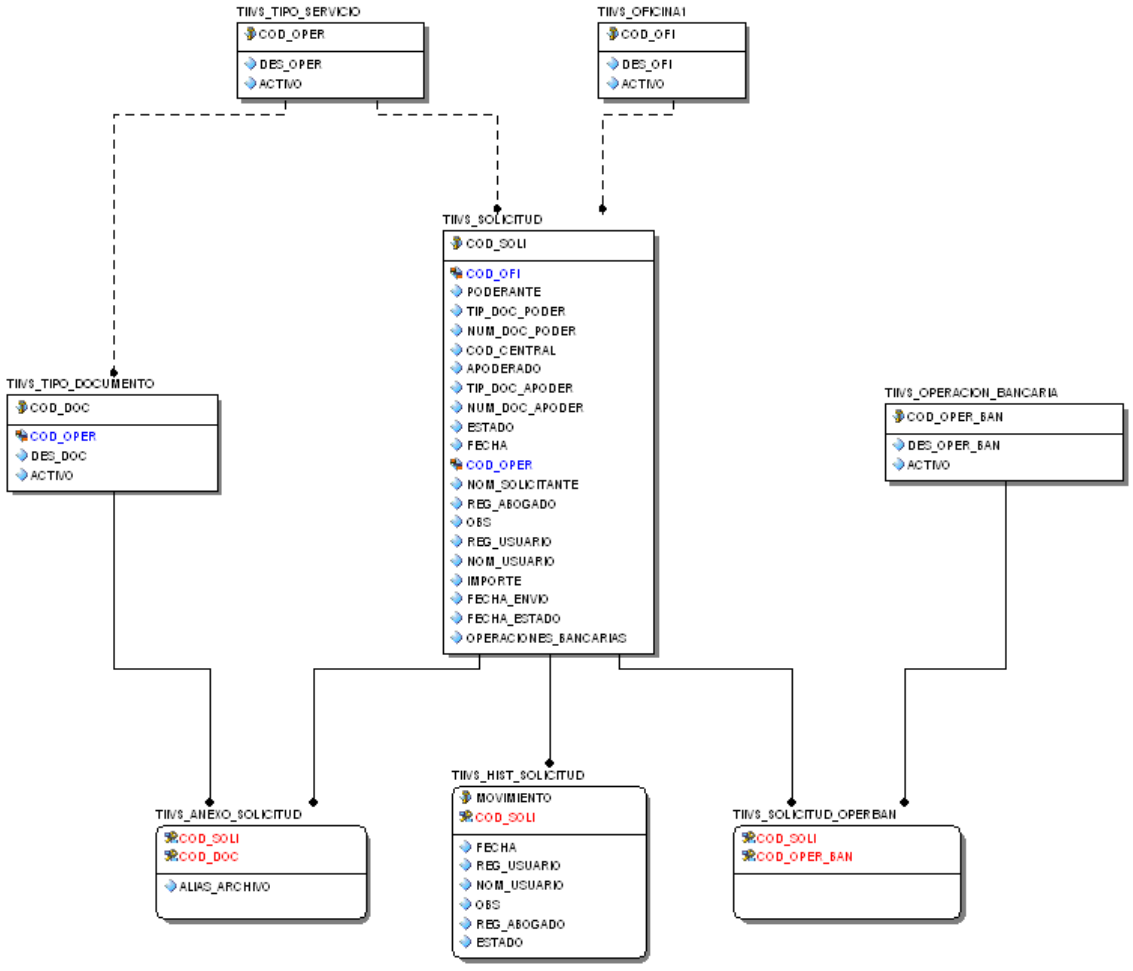
Descripción Funcional

El sistema permite el tráfico de Documentos, requeridos para diversos tipos de Operaciones, entre la Red de Oficinas y el área de Servicios Jurídicos. Dichos documentos son almacenados en formato digital y pueden ser accedidos a través del sistema por las áreas involucradas.

Adicionalmente, el sistema permite administrar los Tipos de Servicios y Documentos a presentar para cada Tipo de Servicio. Así mismo, el sistema incluye un módulo de Configuraciones para Perfiles, Parámetros, etc.

4.2.1 Modelado de negocio del prototipo

Modelo de datos del Sistema Visado de Poderes



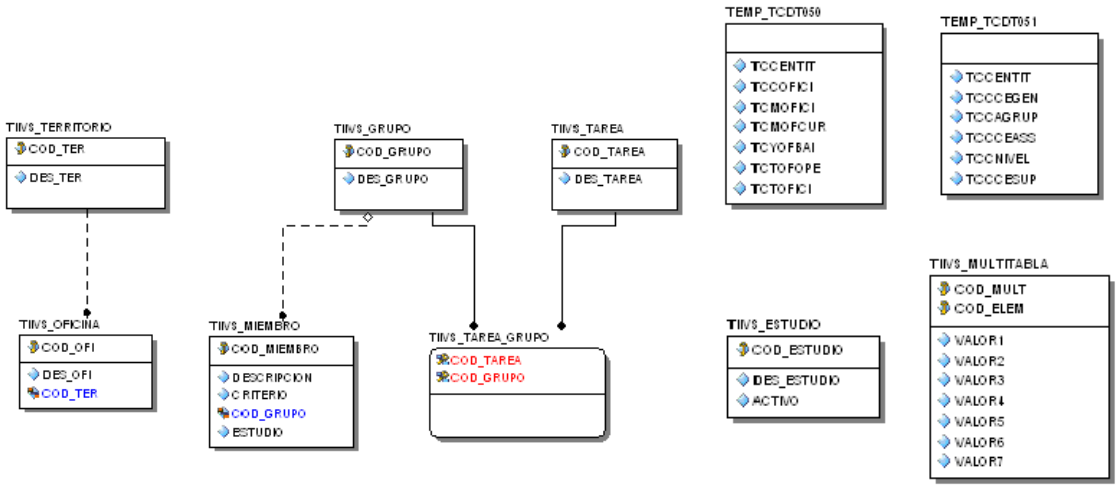


Figura 17 Modelo de datos fuente propia

Prototipos del Sistema Visado de Poderes

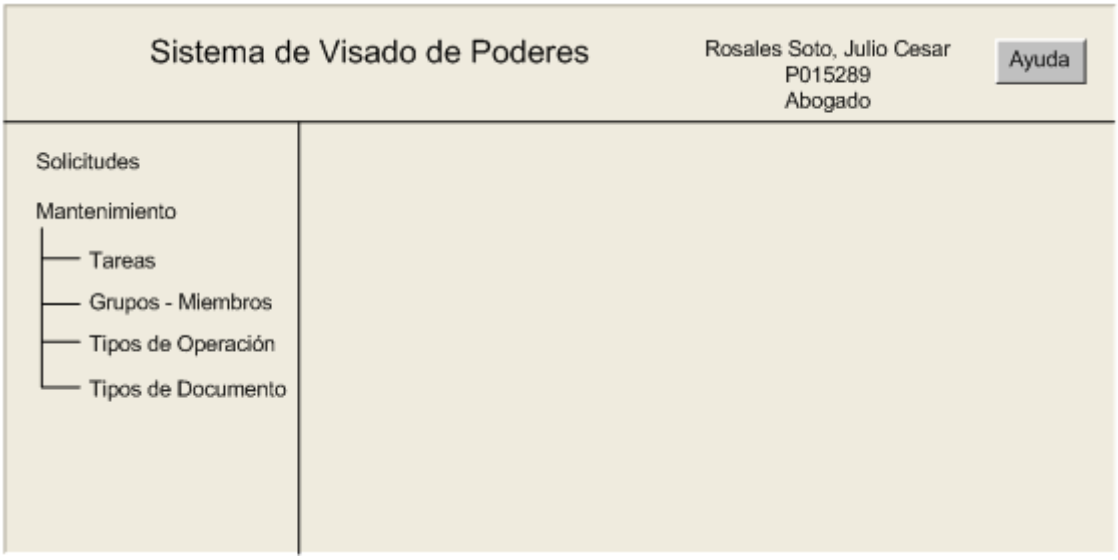
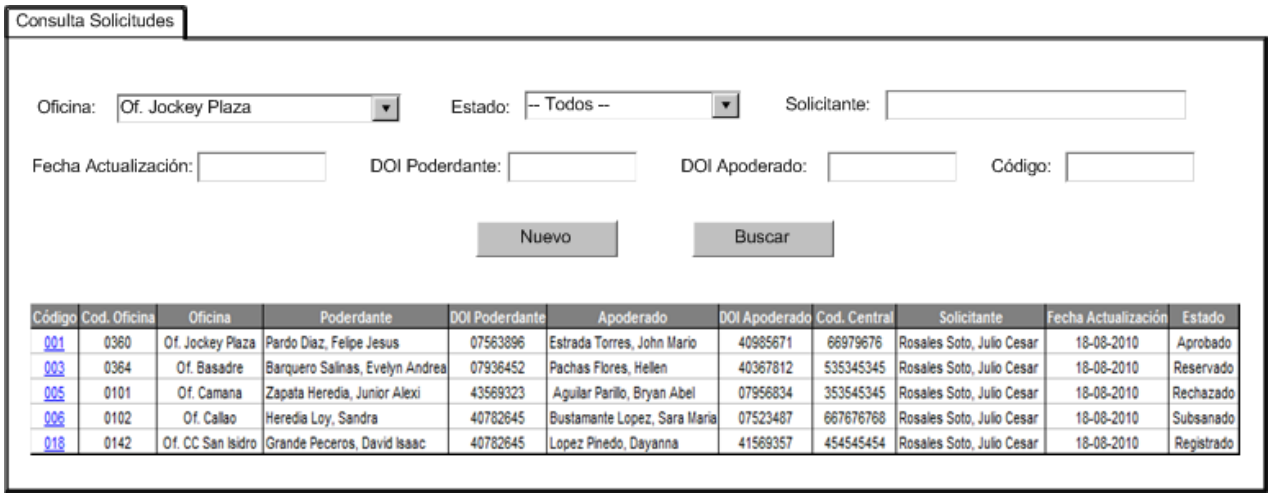
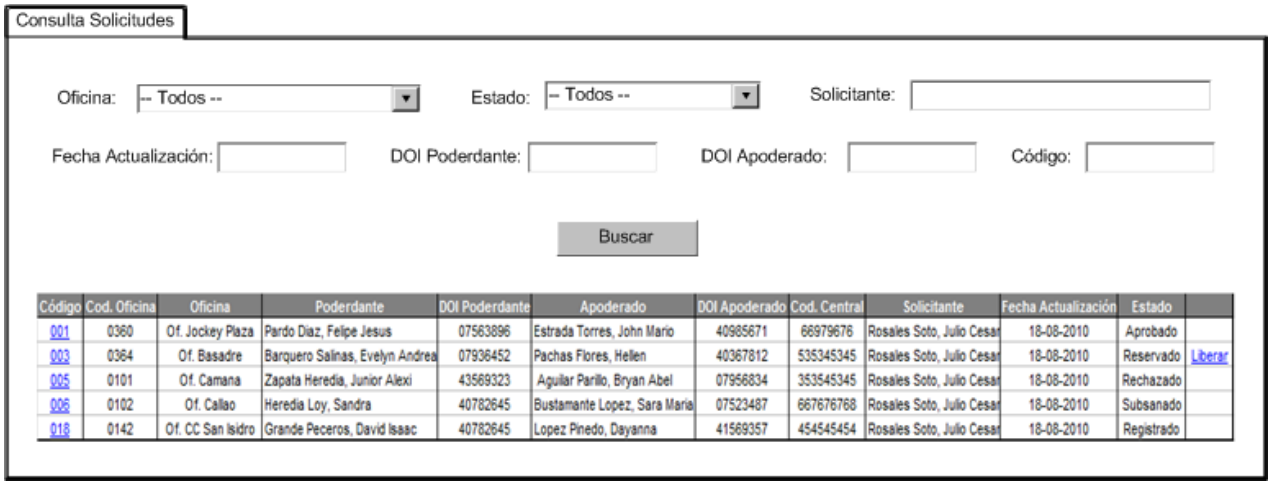


Figura 18 Prototipo – Pantalla Principal fuente propia



Código	Cod. Oficina	Oficina	Poderdante	DOI Poderdante	Apoderado	DOI Apoderado	Cod. Central	Solicitante	Fecha Actualización	Estado
001	0360	Of. Jockey Plaza	Pardo Diaz, Felipe Jesus	07563896	Estrada Torres, John Mario	40985671	66979676	Rosales Soto, Julio Cesar	18-08-2010	Aprobado
003	0364	Of. Basadre	Barquero Salinas, Evelyn Andrea	07936452	Pachas Flores, Hellen	40367812	535345345	Rosales Soto, Julio Cesar	18-08-2010	Reservado
005	0101	Of. Camana	Zapata Heredia, Junior Alexi	43569323	Aguilar Parillo, Bryan Abel	07956834	353545345	Rosales Soto, Julio Cesar	18-08-2010	Rechazado
006	0102	Of. Callao	Heredia Loy, Sandra	40782645	Bustamante Lopez, Sara Maria	07523487	667676768	Rosales Soto, Julio Cesar	18-08-2010	Subsanado
018	0142	Of. CC San Isidro	Grande Peceros, David Isaac	40782645	Lopez Pinedo, Dayanna	41569357	454545454	Rosales Soto, Julio Cesar	18-08-2010	Registrado

Figura 19 Prototipo Bandeja de Solicitudes para el Perfil Administrador fuente propia



Código	Cod. Oficina	Oficina	Poderdante	DOI Poderdante	Apoderado	DOI Apoderado	Cod. Central	Solicitante	Fecha Actualización	Estado	
001	0360	Of. Jockey Plaza	Pardo Diaz, Felipe Jesus	07563896	Estrada Torres, John Mario	40985671	66979676	Rosales Soto, Julio Cesar	18-08-2010	Aprobado	
003	0364	Of. Basadre	Barquero Salinas, Evelyn Andrea	07936452	Pachas Flores, Hellen	40367812	535345345	Rosales Soto, Julio Cesar	18-08-2010	Reservado	Liberar
005	0101	Of. Camana	Zapata Heredia, Junior Alexi	43569323	Aguilar Parillo, Bryan Abel	07956834	353545345	Rosales Soto, Julio Cesar	18-08-2010	Rechazado	
006	0102	Of. Callao	Heredia Loy, Sandra	40782645	Bustamante Lopez, Sara Maria	07523487	667676768	Rosales Soto, Julio Cesar	18-08-2010	Subsanado	
018	0142	Of. CC San Isidro	Grande Peceros, David Isaac	40782645	Lopez Pinedo, Dayanna	41569357	454545454	Rosales Soto, Julio Cesar	18-08-2010	Registrado	

Figura 20 Prototipo Bandeja de Solicitudes para el Perfil Usuario Oficina fuente propia

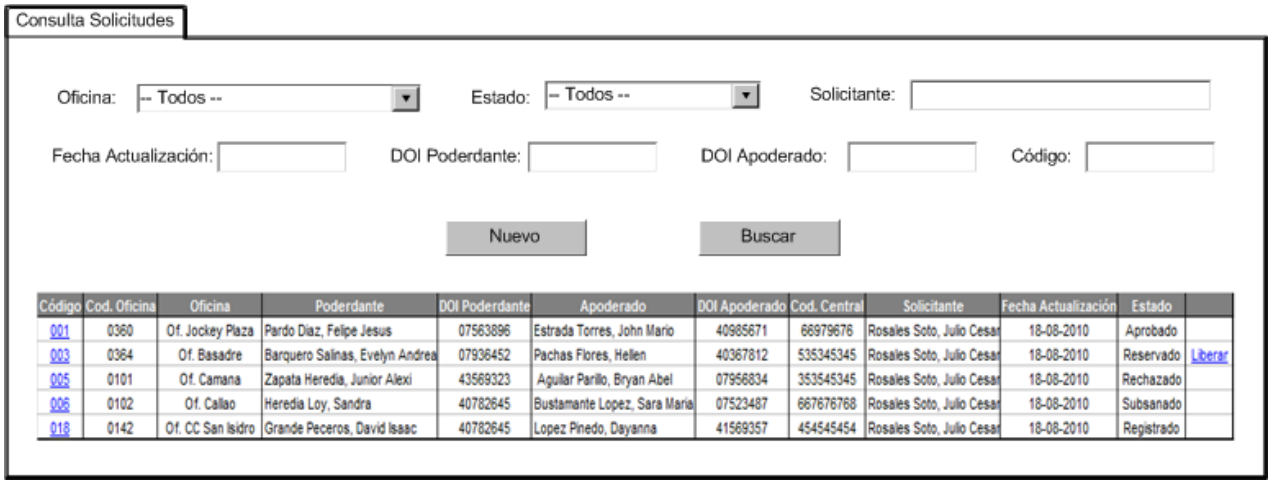


Figura 21 Prototipo Bandeja de Solicitudes para el Perfil Abogado fuente propia

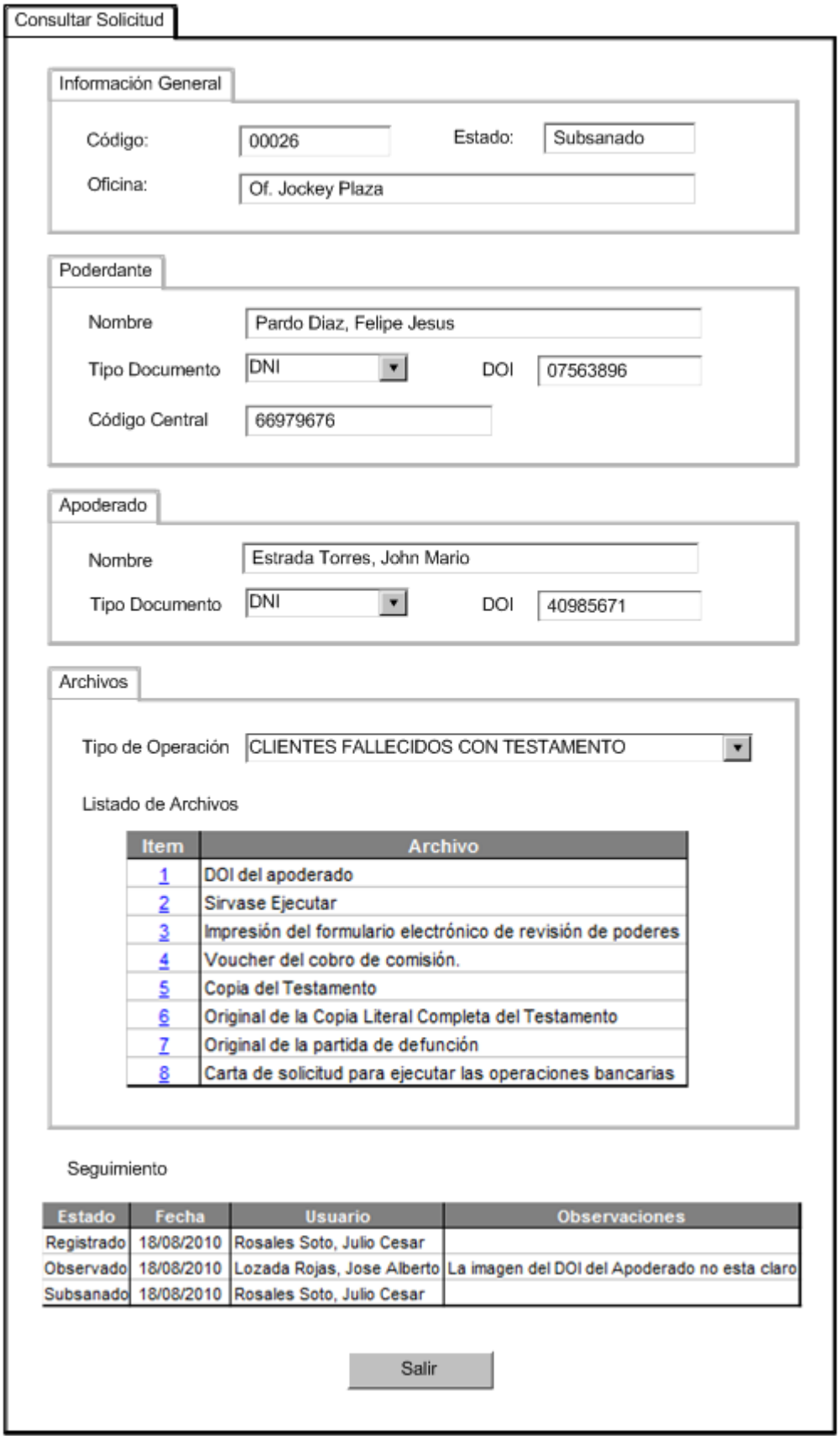


Figura 22 Prototipo para Consultar Solicitud fuente propia

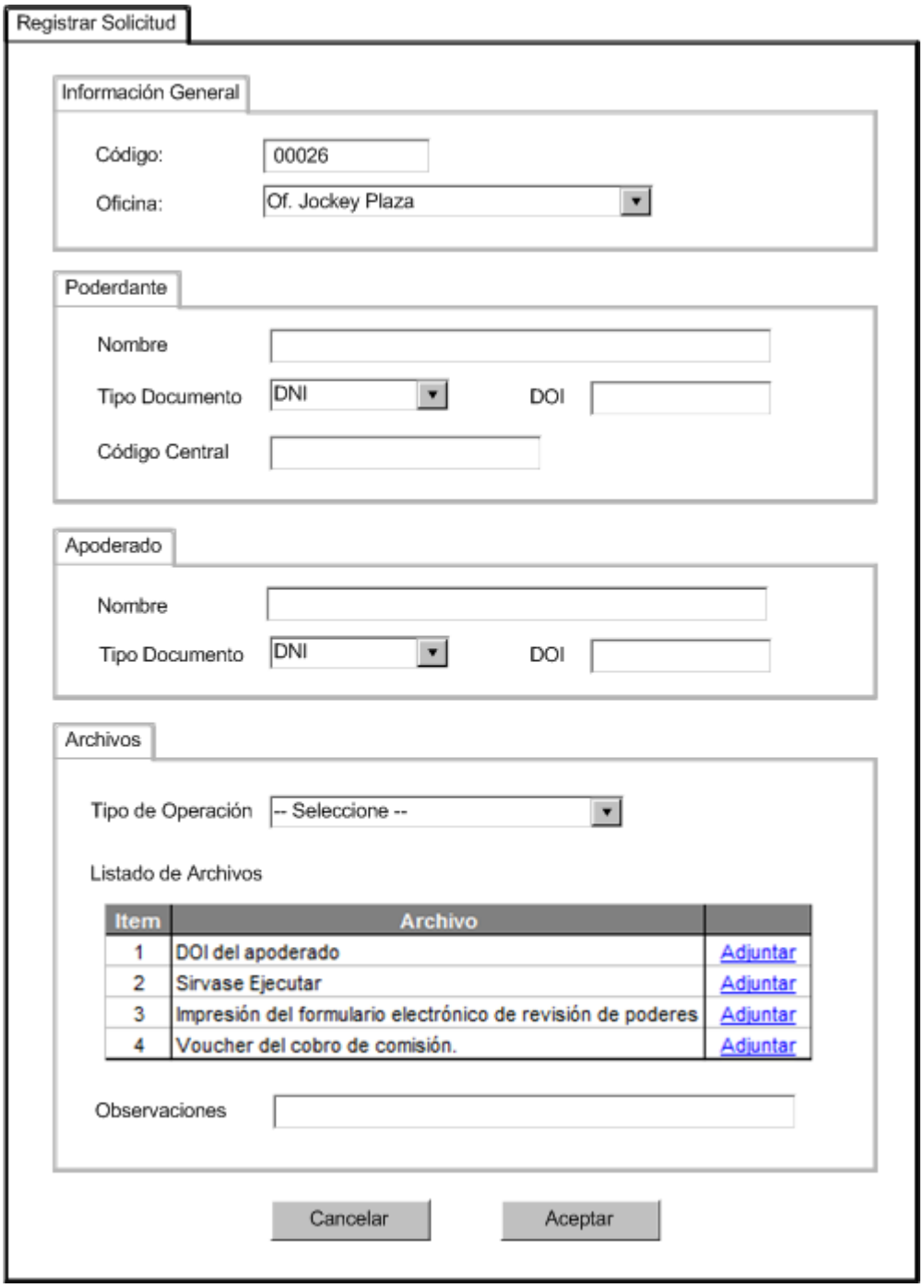


Figura 23 Prototipo para Registrar Solicitud fuente propia

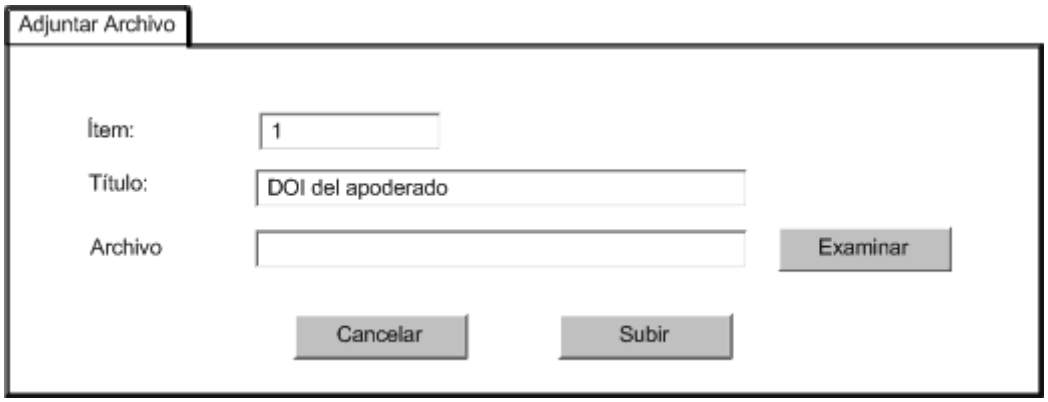


Figura 24 Prototipo para adjuntar Archivo fuente propia

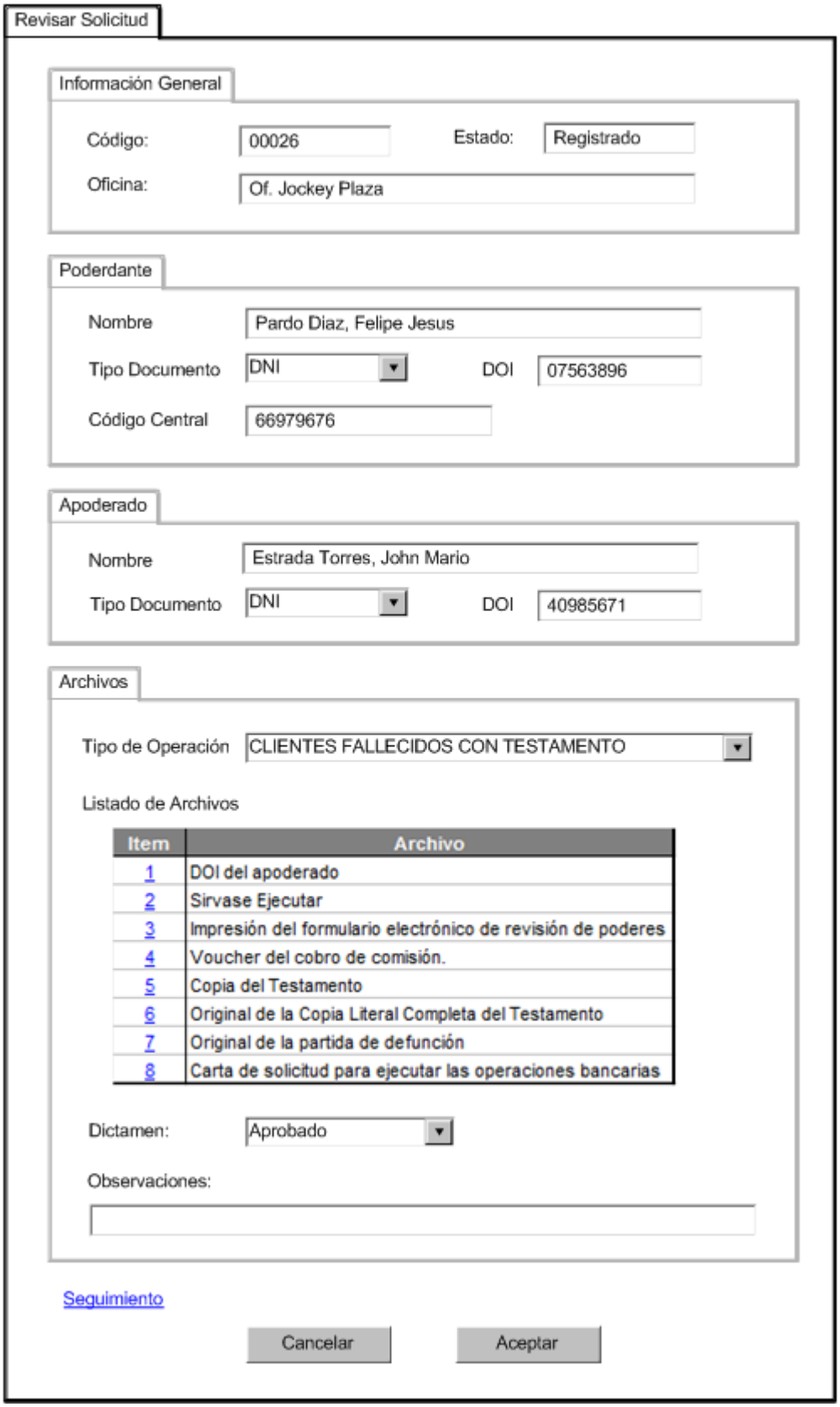


Figura 25 Prototipo para Revisar Solicitud fuente propia

4.2.2 Diagrama de actividades/Flujos de Procesos

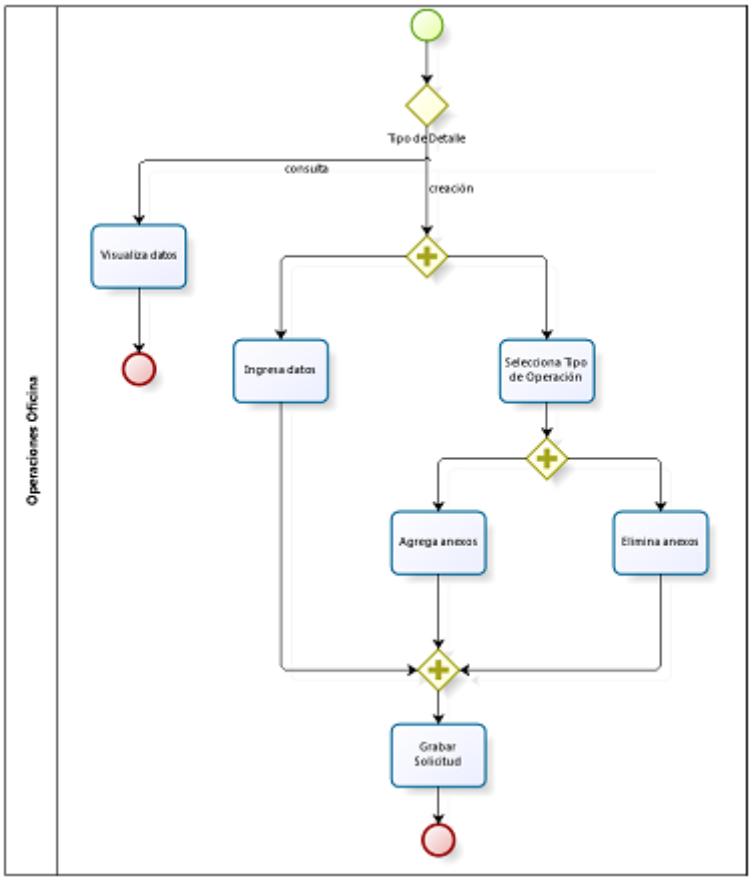


Figura 26 Flujo de Usuario Oficina fuente propia

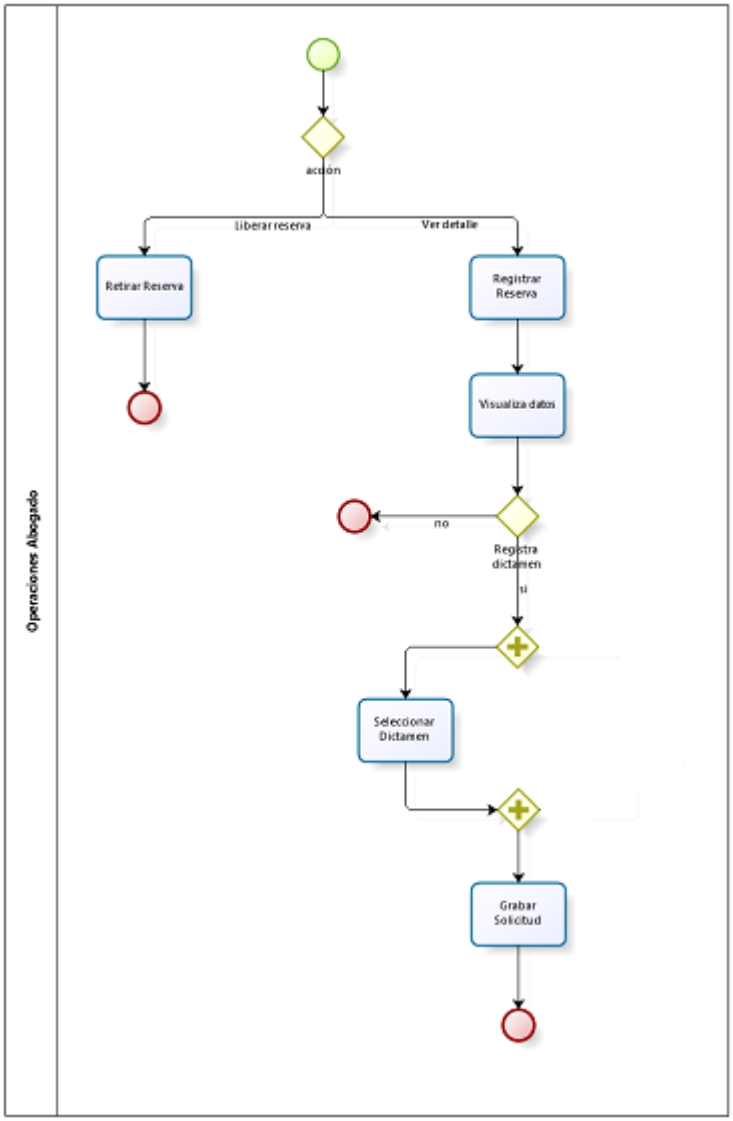


Figura 27 Flujo de Usuario Abogado fuente propia

4.2.3 Diagramas con las especificaciones de los casos de uso

NOMBRE DEL CASO DE USO Acceso al sistema
OBJETIVOS El objetivo principal de este caso de uso es el de validar el acceso al sistema.
INVENTARIO DE ACTORES Usuario intranet.

CARACTERÍSTICAS

PRECONDICIONES El usuario deberá ingresar a través de Espacio Perú.
POSTCONDICIONES

Escenario Primario.

Paso	Acción
1	El usuario presiona el enlace que se encuentra en Espacio Perú.
2	A continuación se validara que el usuario se encuentre registrado como miembro de uno de los grupos configurados en el sistema.
3	De pertenecer a unos de los grupos configurados en el sistema se re direccionara a la bandeja principal del sistema habilitándose las opciones correspondientes al grupo que pertenece.
4	De no pertenecer a ninguno de los grupos configurados en el sistema se re direccionara a una pantalla con un mensaje indicando que el usuario no se encuentra registrado.

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

NOMBRE DEL CASO DE USO Grupos - Miembros
OBJETIVOS El objetivo principal de este caso de uso es el de registrar a los usuarios del sistema.
INVENTARIO DE ACTORES Usuario Administrador.

CARACTERÍSTICAS

PRECONDICIONES El usuario administrador deberá estar registrado como tal.
POSTCONDICIONES

Escenario Primario.

Paso	Acción
1	El usuario administrador accederá atreves del menú mantenimiento opción "Grupos – Miembros".
2	El usuario administrador podrá registrar nuevos usuarios así como editar aquellos que se encuentren registrados, mediante el botón "Nuevo" y los enlaces "Editar" y "Eliminar". Así mismo podrá buscar a los usuarios por grupo a través del filtro "Grupo" y el botón "Buscar".
3	Si selecciona el botón "Nuevo", al presionar este botón, se mostrara una ventana donde se ingresaran los datos: ✓ Criterio: Obligatorio y contiene la lista de criterios (Categoría, Oficina, Registro) ✓ Código: Obligatorio ✓ Descripción: Obligatorio
4	Al presionar el botón "Aceptar", se graba la información y se regresa a la pantalla de "Grupos - Usuarios".

	Para editar un miembro, se va a la misma pantalla y escoge la opción “Editar” de la fila que se quiere editar
5	Si selecciona el enlace “Editar”, al presionar esta opción, se abrirá una ventana de edición donde solo se pueda editar la descripción. Se presiona “Aceptar” para grabar y regresar a la pantalla anterior.
6	Si selecciona el enlace “Eliminar”, al presionar esta opción, se eliminará el registro seleccionado.

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

NOMBRE DEL CASO DE USO Consulta de Solicitudes
OBJETIVOS El objetivo principal de este caso de uso es el de consultar las solicitudes registradas en el sistema.
INVENTARIO DE ACTORES Todos los usuarios registrados en el sistema.

CARACTERÍSTICAS

PRECONDICIONES
POSTCONDICIONES

Escenario Primario.

Paso	Acción
1	El usuario accederá a la interfaz de consulta de Solicitudes a través de la opción del menú “Solicitudes→ Bandeja”.

2	A continuación se mostrara una interfaz con filtros de búsqueda y una grilla de resultados, de acuerdo a lo permisos del perfil al que pertenezca el usuario.
3	El usuario llena los criterios de búsqueda y da clic en el botón “Buscar”.
4	El sistema cargara la grilla de resultados con las coincidencias de acuerdo a los criterios establecidos por el usuario mediante los filtros de búsqueda o un mensaje de “no se encontró registros…”, en caso no haya coincidencias.
5	El usuario da clic en el “Número de Solicitud” de la grilla de resultados del registro seleccionado, el cual re direccionara al interfaz de Solicitudes para poder ver a detalle la Solicitud.

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

NOMBRE DEL CASO DE USO Registro de Solicitudes
OBJETIVOS El objetivo principal de este caso de uso es el de registrar Solicitudes a visar en el sistema.
INVENTARIO DE ACTORES Usuario Administrador y Oficina.

CARACTERÍSTICAS

PRECONDICIONES
POSTCONDICIONES

Escenario Primario.

Paso	Acción

1	El usuario da clic en el botón “Nuevo” de la interfaz de Consulta de Solicitudes
2	A continuación se mostrara la interfaz para Registrar Solicitudes con los campos requeridos por la Solicitud.
3	El usuario da clic en el botón “Guardar” de la interfaz Registrar Solicitudes, mediante el cual procederá a registrar la Solicitud en el sistema.
4	A continuación se mostrara un mensaje de confirmación y se re direccionara a la interfaz de Consulta de Solicitudes.
5	El usuario da clic en el botón “Enviar a SSJJ” de la interfaz Registrar Solicitudes, mediante el cual se procederá a asignar la solicitud a un determinado Abogado del área legal mediante la asignación automática.
6	El usuario da clic en el botón “Cancelar” de la interfaz Registrar Solicitudes, mediante el cual se retornara direccionara a la interfaz de Consulta de Solicitudes.

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

NOMBRE DEL CASO DE USO Reserva de solicitudes
OBJETIVOS El objetivo principal de este caso de uso es el de dar inicio a al proceso de validación de las Solicitudes a visar en el sistema.
INVENTARIO DE ACTORES Usuario Abogado.

CARACTERÍSTICAS

PRECONDICIONES
POSTCONDICIONES

Escenario Primario.

Paso	Acción
1	El usuario selecciona un registro de la grilla de resultados de la interfaz de Consulta de Solicitudes.
2	El usuario da clic en el “Número de Solicitud” de la grilla de resultados del registro seleccionado, el cual re direccionara al interfaz de Solicitudes para poder ver a detalle la Solicitud, lo cual es suficiente para poder reservar dicha solicitud.

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

NOMBRE DEL CASO DE USO Revisión de Solicitudes
OBJETIVOS El objetivo principal de este caso de uso es el de dar finalizar el proceso de validación de las Solicitudes a visar en el sistema.
INVENTARIO DE ACTORES Usuario Abogado.

CARACTERÍSTICAS

PRECONDICIONES
POSTCONDICIONES

Escenario Primario.

Paso	Acción

1	El usuario s
2	
3	
4	

REGLAS DE NEGOCIO:

Descripción General

REQUERIMIENTOS NO FUNCIONALES N/A
NOTAS PARA IMPLEMENTACIÓN N/A

4.2.3.1 Diagrama de estado

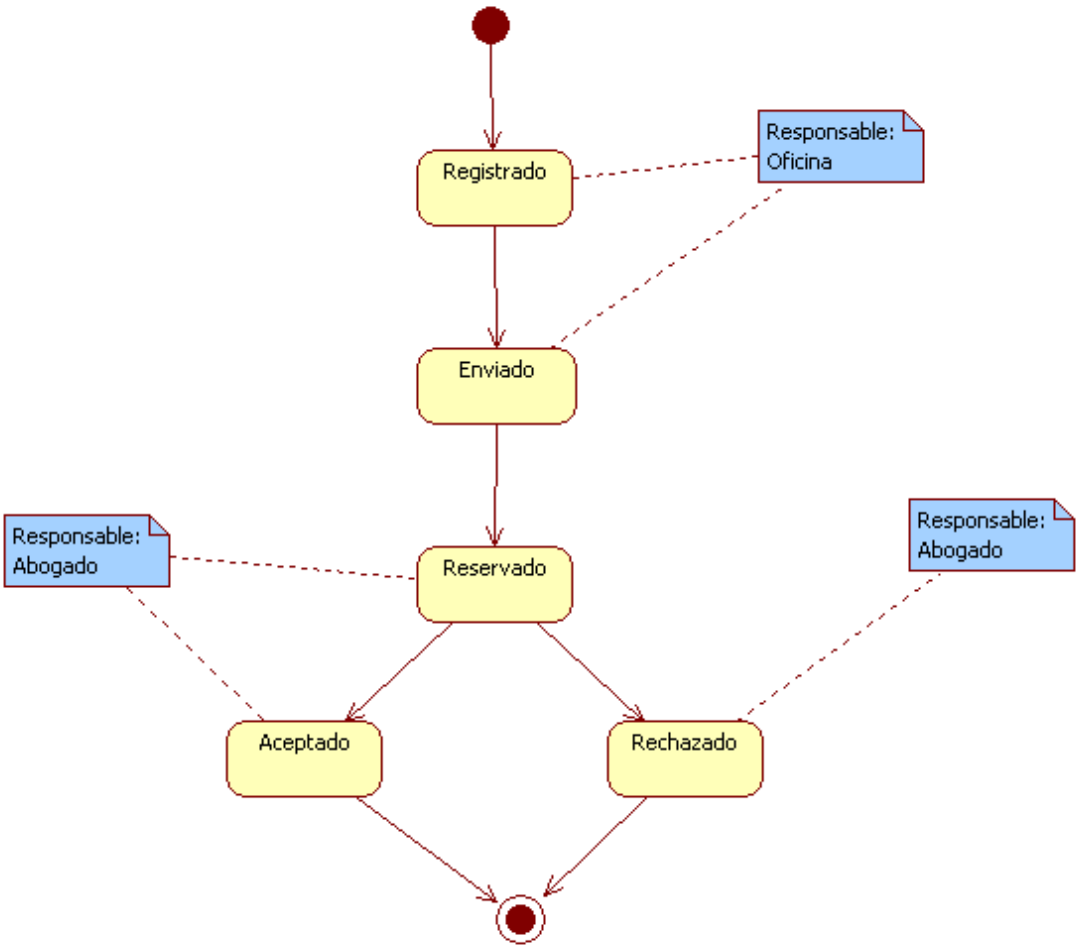


Figura 28 Diagrama de Estados fuente propia

4.2.3.2 Diagrama de secuencia

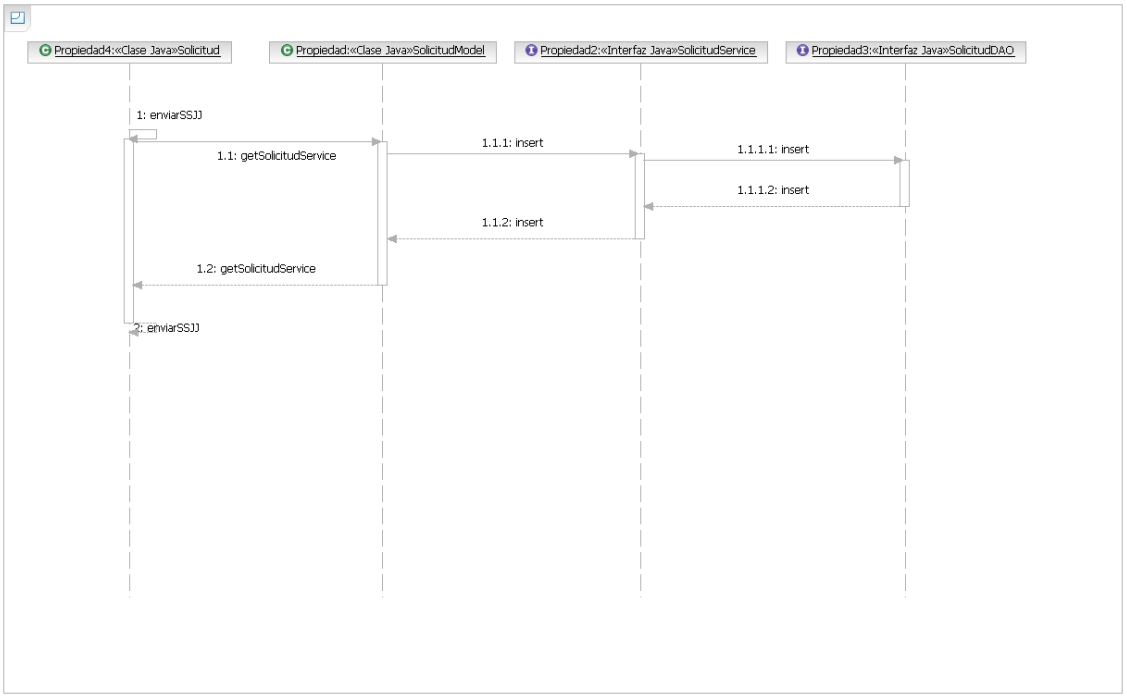
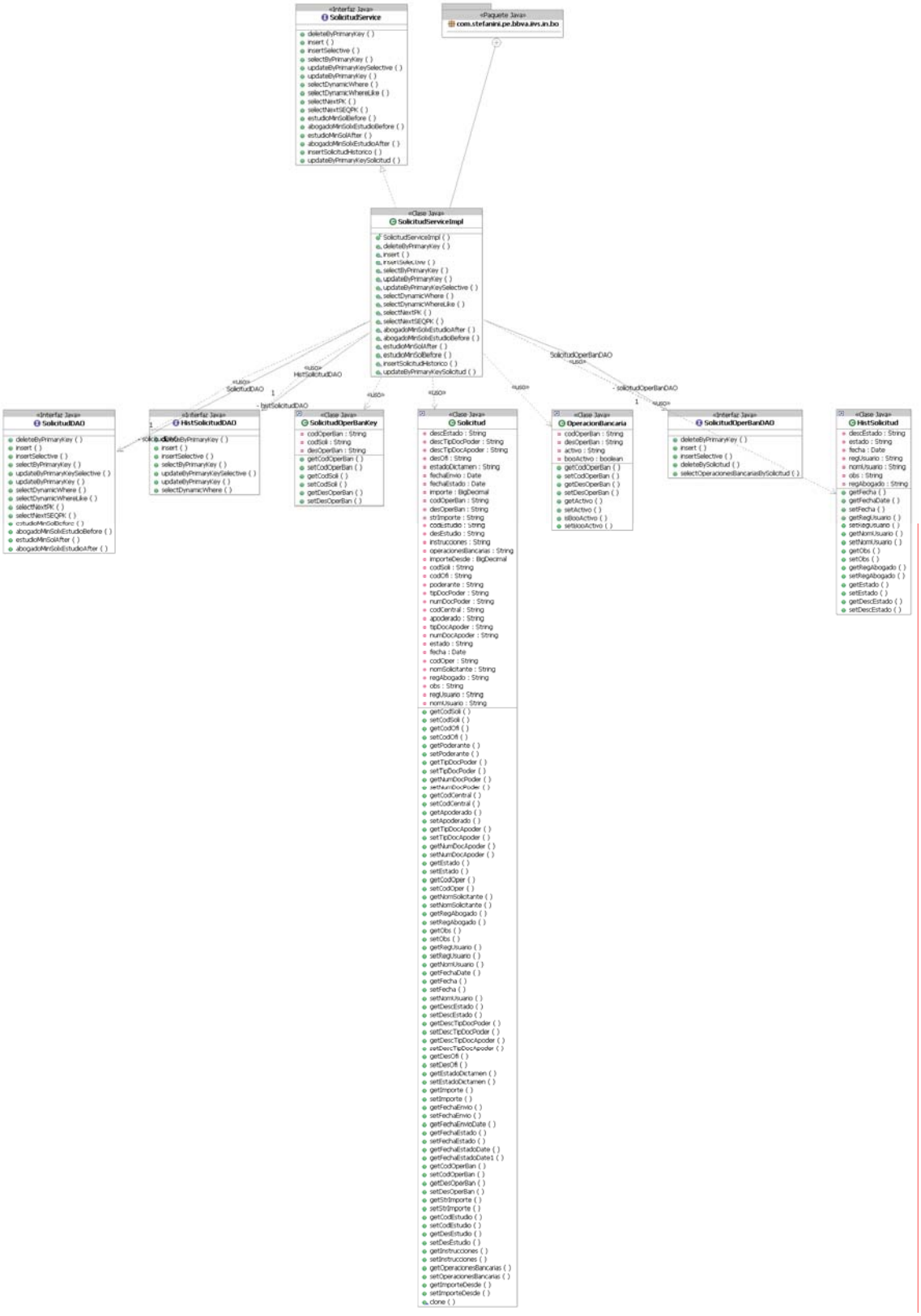


Figura 29 Diagrama Secuencia Enviar a SSJJ fuente propia

4.2.3.3 Diagrama de clases



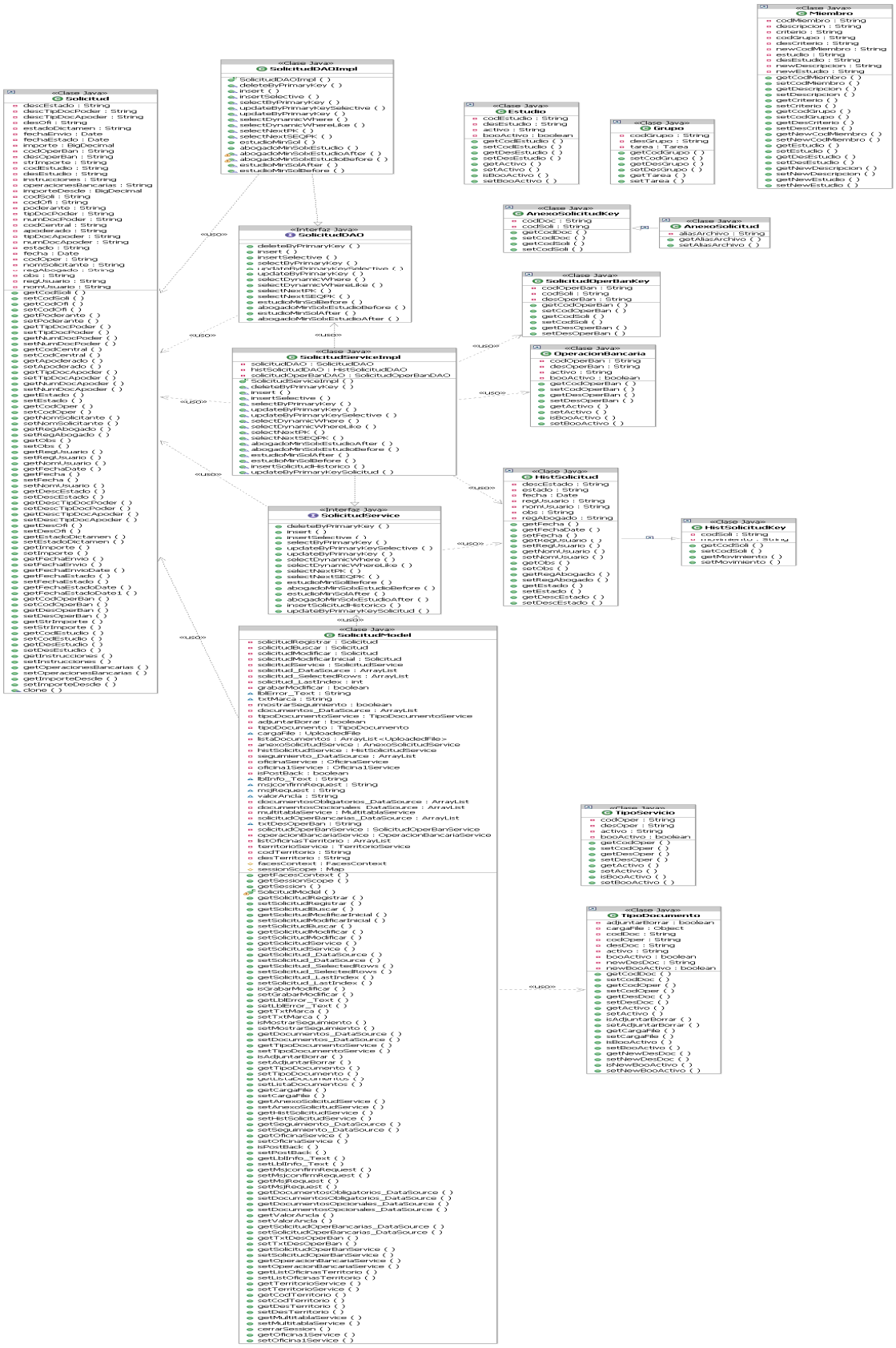


Figura 30 Diagrama de Clases fuente propia

4.2.3.4 Consideraciones sobre el ambiente de desarrollo (entorno de desarrollo utilizado, archivos de datos, alcances y limitaciones del sistema, módulos del sistema y clases más importantes)

Consideraciones a tener en cuenta en el desarrollo del Sistema de Visado de Poderes:

Entorno de desarrollo:

IBM Rational Software Architect (RSA) o **IBM Rational Application Developer** (RAD)

Java Development Kit (**JDK**) 1.5

Servidor de aplicaciones:

IBM WebSphere Application Server (**WAS**, servidor de aplicaciones WebSphere)

Sistema de gestión de base de datos:

Oracle Database 10g Release 2 Enterprise Edition

4.2.3.5 Módulos del sistema

Actualmente el sistema se divide en dos módulos el modulo de Administración Solicitudes y el modulo de Mantenimientos.

El modulo de Administración Solicitudes:

El cual permite consultar las solicitudes existentes y registrar nuevas solicitudes, así mismo la reserva y revisión de solicitudes asignadas a los distintos abogados del área legal.

El modulo de Mantenimientos:

El cual permite registrar la configuración necesaria para el correcto funcionamiento del sistema de visado de poderes.

4.2.3.6 Requerimiento mínimo de hardware y software

Requerimientos mínimos para el correcto funcionamiento del Sistema de Visado de Poderes

Consideraciones a tener en cuenta al implementar el Sistema de Visado de Poderes:

Servidor de aplicaciones:

IBM WebSphere Application Server (**WAS**, servidor de aplicaciones WebSphere)

Java Development Kit (**JDK**) 1.5

Sistema de gestión de base de datos:

Oracle Database 10g Release 2 Enterprise Edition

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- ✓ El algoritmo del banquero en ciertas condiciones (exclusión mutua y recursos limitados) puede ser de gran utilidad, en condiciones donde las desventajas de este algoritmo pasen desapercibidas o sean minimizadas, el proceso de asignación de recursos seria optimo.
- ✓ Si se tiene la certeza que el sistema donde se va utilizar el algoritmo del banquero siempre va estar en un estado seguro se puede omitir la parte del algoritmo de verificación estado que utiliza el algoritmo del banquero.
- ✓ El algoritmo del banquero detecta posibles interbloqueos a través del algoritmo de seguridad que utiliza.
- ✓ Los algoritmos de tratamiento de interbloqueos se podrían clasificar de acuerdo a la estrategia a utilizar si se va prevenir, evitar, detectar o recuperar.
- ✓ De acuerdo a la necesidad y a las condiciones presentes del negocio, donde se va implementar estos tipo de algoritmos de de tratamiento de interbloqueos, se debe elegir la estrategia a utilizar.

5.2 Recomendaciones

- ✓ Tener bien claro la diferencia entre algoritmos de asignación de recursos con los algoritmos de concurrencia dentro del área de estudio de los sistemas operativos.
- ✓ Se recomienda utilizar el algoritmo del banquero acompañado de un algoritmo de concurrencia, o en el caso de la implantación en el lenguaje de programación Java se puede utilizar como alternativas hilos.
- ✓ Se recomienda para la implementación de sistema que trabaja con repositorios de archivos un content manager que se encargue de trafico de los archivos requeridos.
- ✓ Se debe tener en cuenta las condiciones o reglas del negocio donde se va implementar el sistema para poder determinar los parámetros de entrada del algoritmo del banquero.
- ✓ En el futuro se utilizaran todos los parámetros de entrada del algoritmo del banquero siempre y cuando las condiciones o reglas del negocio lo ameriten.

REFERENCIAS BIBLIOGRÁFICAS.

Libros

[1]A. S. Tanenbaum. 1996. Sistemas Operativos, 2ª Edición, Modernos. México: Prentice Hall Hispanoamericana.

[2]Deitel, H. M. 1993. Sistemas Operativos, 2ª Edición, Boston: Addison-Wesley.

[3]Milenkovic M. Sistemas Operativos. 1994. Conceptos y Diseño, 2ª Edición, Madrid: McGraw-Hill.

[4]Stallings, W. 1993. Computer Organization and Architecture, 3ª Edición. New York: Macmillan.

Artículos de Revistas Científicas

[5]Isabel Medrano Corrales y Luis Suárez Samaniego. 2005. Gestión de archivos particulares en la era digital. El Profesional de la Información, v. 14, n. 6, 442-448.

Fuentes electrónicas

[6]Algoritmo del banquero. (n.d.). Obtenida el 20 de marzo de 2011, de http://es.wikipedia.org/wiki/Asignaci%C3%B3n_de_recursos/

[7]Asignación de recursos. (n.d.). Obtenida el 09 de marzo de 2011, de http://es.wikipedia.org/wiki/Asignaci%C3%B3n_de_recursos/

[8]Asignación de recursos. (n.d.). Obtenida el 08 de marzo de 2011, de http://www.eco-finanzas.com/diccionario/A/ASIGNACION_DE_RECURSOS.htm

[9]Bloqueo mutuo. (n.d.). Obtenida el 09 de marzo de 2011, de http://es.wikipedia.org/wiki/Bloqueo_mutuo

[10]Ciencias de la computación. (n.d.). Obtenida el 18 de junio de 2011, de http://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n

[11]Gestión empresarial orientada al valor del cliente como fuente de ventaja competitiva. (n.d.). Propuesta de un modelo explicativo. Obtenida el 08 de marzo de 2011, de http://biblioteca.universia.net/html_bura/ficha/params/title/gestion-empresarial-orientada-valor-cliente-como-fuente-ventaja-competitiva-propuesta/id/38995186.html

[12]Java lenguaje de programación. (n.d.). Obtenida el 15 de junio de 2011, de http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[13]Lenguaje Unificado de Modelado. (n.d.). Obtenida el 15 de junio de 2011, de http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

[14]Procesos de un Programa de Gestión de Documental. (n.d.). Obtenida el 09 de marzo de 2011, de http://elcacique87.blogspot.com/2008_06_01_archive.html

[15]Sistema De Trámite Documentario. (n.d.). Obtenida el 10 de marzo de 2011, de http://www.buenastareas.com/ensayos/Sistema-De-Tramite-Documentario/540025.html

[16]Sistemas Operativos. (2002). Obtenida el 09 de mayo de 2011, Universidad Nacional del Nordeste página web de Departamento de informática: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/CINTIN02.htm

[17]Sistemas Operativos. (1998). Obtenida el 09 de mayo de 2011, Universidad de Jaen Departamento de informática: http://wwwdi.ujaen.es/~lina/TemasSO/INTERBLOQUEOS/3y4CondicionesNecesariasyEstrategiasdeinterbloqueo.htm

ANEXOS